ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ
ՀԱՄԱԼՍԱՐԱՆ

Լ. ԽԱՉՈՅԱՆ | Ա. ԱՆԴՐԵԱՍՅԱՆ | Ա. ԱՌԱՔԵԼՅԱՆ
Վ. ՁԱՔԱՐՅԱՆ | Ա. ՄԱՆՈՒԿՅԱՆ | Զ. ՆԱԶԱՐՅԱՆ

ԾՐԱԳՐԱՎՈՐՈՒՄ

ԼԵՋՎՈՎ

{ ԽՆԴԻՐՆԵՐԻ ԺՈՂՈՎԱԾՈՒ }

ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Լ. Օ. Խաչոյան, Ա. Ս. Անդրեասյան
Ա. Հ. Առաքելյան, Վ. Ա. Զաքարյան
Ա. Գ. Մանուկյան, Զ. Ա. Նազարյան

ԾՐԱԳՐԱՎՈՐՈՒՄ C++ ԼԵԶՎՈՎ

(խնդիրների ժողովածու)

ԵՐԵՎԱՆ
ԵՊՀ ՀՐԱՏԱՐԱԿՉՈՒԹՅՈՒՆ
2019

ՀՏԴ 004.42(076.1)
ԳՄԴ 32.973-018ց7
Ծ 892

Հրատարակության է երաշխավորել
ԵՊՀ ինֆորմատիկայի և կիրառական մաթեմատիկայի
ֆակուլտետի գիտական խորհուրդը:

Լ. Օ. Խաչոյան, Ա. Ս. Անդրեասյան, Ա. Հ. Առաքելյան,
Վ. Ա. Զաքարյան, Ա. Գ. Մանուկյան, Զ. Ա. Նազարյան
Ծ 892 Ծրագրավորման խնդիրների ժողովածու: Ուսումնամեթոդական ձեռնարկ: Եր., ԵՊՀ հրատ., 2019, 234 էջ:

Խնդիրների ժողովածուն, որը կազմվել է Երևանի պետական համալսարանի «Ծրագրավորման և ինֆորմացիոն տեխնոլոգիաների» ամբիոնի աշխատակիցների ջանքերով, ներառում է երկար տարիների ընթացքում Ինֆորմատիկայի և կիրառական մաթեմատիկայի ֆակուլտետի «ԷՀՄ և ծրագրավորում» առարկայի դասախոսությունների և գործնական պարապմունքների դասավանդման ժամանակ օգտագործված բազմաթիվ ու բազմաբնույթ խնդիրներ, վարժություններ, թեստեր: Դրանք բաշխված են ըստ թեմաների, որոնցից յուրաքանչյուրը սկսվում է թեմայի համառոտ տեսական նկարագրությամբ և որոշ առաջադրանքների լուծումներով ու դրանց վերաբերյալ մեթոդական ցուցումներով:

Խնդիրների ժողովածուն նախատեսված է ԵՊՀ ինֆորմատիկայի և կիրառական մաթեմատիկայի ֆակուլտետի ուսանողների համար: Այն կարող է օգտակար լինել նաև համալսարանի բնագիտական ֆակուլտետների, ինչպես նաև այլ բուհերի այն ֆակուլտետների ուսանողների համար, ովքեր ուսումնասիրում են C++ ծրագրավորման լեզուն (ինչպես նաև այլ ծրագրավորման լեզուներ):

Հեղինակները երախտագիտություն են հայտնում «Ծրագրավորման և ինֆորմացիոն տեխնոլոգիաների» ամբիոնի աշխատակիցներին, ովքեր իրենց տրամադրած խնդիրներով և արժեքավոր դիտողություններով նպաստել են ժողովածուի հրատարակմանը:

ՀՏԴ 004.42(076.1)
ԳՄԴ 32.973-018ց7

ISBN 987-5-8084-2417-3

Բովանդակություն

1. Թվարկության համակարգեր, կոդեր

Թվարկության համակարգ անվանում են թվերի գրության և ներկայացման կանոնների համախումբը: Նիշերը, որոնք օգտագործվում են թվարկության համակարգում, անվանում են թվանշաններ: Թվարկության համակարգերը լինում են դիրքային և ոչ դիրքային: Ոչ դիրքային համակարգերում թվերի գրառման մեջ թվանշանի արտահայտած արժեքը կախված չէ նրա դիրքից: Ոչ դիրքային համակարգի օրինակ է հռոմեական համակարգը (XXIII=X+X+I+I+I=10+10+1+1+1=23):

Դիրքային համակարգերում թվանշանով տրվող արժեքը կախված է նրա դիրքից: Համընդհանուր կիրառություն ստացած 10-ական համակարգը թվարկության դիրքային համակարգ է, որտեղ օգտագործվում են արաբական թվանշանները: Օրինակ, 3437 գրառումը ներկայացնում է 3000+400+30+7 արժեքով թվի կրճատ և հարմար գրելաձևը, որտեղ պարզ երևում է թվանշանների՝ դիրքից կախված տարբեր արժեքներ ներկայացնելու փաստը:

<u>Թվարկության դիրքային համակարգեր</u>

Թվարկության 10-ական համակարգն ունի տասը թվանշան՝ 0,1,...,9: Համակարգի հիմքը տասն է և գրառվում է 10 տեսքով: Այս համակարգում ցանկացած ամբողջ կամ իրական թիվ գրվում է թվանշանների հաջորդականության միջոցով: Այն կարող է պարունակել տասնորդական ստորակետ (կետ), իսկ առջևում - կամ + (որը սովորաբար բաց է թողնվում) նշան: 10-ական թվի արժեքը հավասար է իր թվանշանների և 10-ի համապատասխան աստիճանների արտադրյալների գումարին: Օրինակ՝

$$2328{,}156 = 2 \cdot 10^3 + 3 \cdot 10^2 + 2 \cdot 10^1 + 8 \cdot 10^0 + 1 \cdot 10^{-1} + 5 \cdot 10^{-2} + 6 \cdot 10^{-3}$$

Այս սկզբունքով ներկայացնենք թվարկության p-ական համակարգը, որտեղ p-ն 1-ից մեծ բնական թիվ է: Թվարկության p-ական համակարգն ունի p հատ թվանշան՝ 0,1,…,p−1 արժեքներով: Համակարգի հիմքը p-ն է, գրվում է 10 տեսքով (կարդացվում է մեկ զրո) և այս համակարգում կատարում է տասի ֆունկցիաները: Եթե $p \leq 10$, ապա որպես թվանշաններ օգտագործվում են 0,1,…,p−1 միանիշ թվերը, իսկ $p > 10$ դեպքում՝ 0,1,...,9,A,B,… նիշերը, որտեղ A-ն 10 արժեքով թվանշանն է, B-ն՝ 11 արժեքով և այլն՝ մինչև p−1 արժեքով հերթական տառը: Օրինակներ՝

- 8-ական համակարգի (p=8) թվանշաններն են՝

0,1,2,3,4,5,6,7

- 13-ական համակարգի (p=13) թվանշաններն են՝

0,1,2,3,4,5,6,7,8,9,A,B,C

Ցանկացած վերջավոր գրառում ունեցող T թիվ p-ական համակարգում կարելի է ներկայացնել հետևյալ տեսքով՝

$a_n \dots a_1 a_0 , a_{-1} \dots a_{-k}$, որտեղ $a_i \in \{0,1,\dots,p-1\}$, $-k \leq i \leq n$: Այդ թվի արժեքն է՝

$$T = a_n \cdot p^n + \cdots + a_1 \cdot p^1 + a_0 \cdot p^0 + a_{-1} \cdot p^{-1} + \cdots + a_{-k} \cdot p^{-k}$$

Միարժեքության համար T թիվը p-ական համակարգում նշանակենք T_p : Եթե p=10, ապա ինդեքսը կարելի է բաց թողնել: Օրինակ՝ 10111_2-ը 2-ական համակարգի թիվ է, 10111-ը՝ 10-ական համակարգի, 3204_5–ը՝ 5-ական համակարգի, իսկ $B19AF_{16}$-ը՝ 16-ական համակարգի: Մեզ հետաքրքրում են 2, 8 և 16 հիմքերով համակարգերը:

Թվարկության 2-ական համակարգի թվանշաններն են 0-ն և 1-ը: Համակարգի հիմքը երկուսն է և գրվում է 10 տեսքով (կարդացվում է մեկ զրո): 2-ական թվի օրինակ՝

$$11010{,}11_2 = 1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 + 1\cdot 2^{-1} + 1\cdot 2^{-2} = 16 + 8 + 2 + 0{,}5 + 0{,}25 = 26{,}75_{10}$$

Նմանապես՝ 8-ական համակարգի հիմքը ութն է, իսկ թվանշաններն են 0,1,...,7-ը: Օրինակ՝

$$1207{,}14_8 = 1\cdot 8^3 + 2\cdot 8^2 + 0\cdot 8^1 + 7\cdot 8^0 + 1\cdot 8^{-1} + 4\cdot 8^{-2} = 512 + 128 + 7 + 0{,}125 + 0{,}0625 = 647{,}1875_{10} \approx 647{,}19_{10}$$

Թվարկության 16-ական համակարգն ունի 16 հատ թվանշան՝ 0,1,...,9,A,B,C, D,E,F: Այստեղ A,B,C,D,E,F տառերը համապատասխանաբար 10,11,12,13,14, 15 արժեքներով նիշերն են: Համակարգի հիմքը տասնվեցն է և գրվում է 10 տեսքով (կարդացվում է մեկ զրո): 16-ական թվի օրինակ՝

$$\mathrm{A3F,B4}_{16} = 10\cdot 16^2 + 3\cdot 16^1 + 15\cdot 16^0 + 11\cdot 16^{-1} + 4\cdot 16^{-2} = 2560 + 48 + 15 + 0{,}6875 + 0{,}015625 = 2623{,}703125_{10} \approx 2623{,}703_{10}$$

Քանի որ $10_p = 1\cdot p^1 + 0\cdot p^0 = p$, ուստի p-ական համակարգում p-ն գրառվում է 10 տեսքով: p-ական համակարգի թվերի հետ թվաբանական գործողություններ կատարելու համար պետք է իմանալ այդ գործողությունների աղյուսակները տվյալ համակարգում: Գործողությունները կատարվում են նույն ալգորիթմներով, ինչպես որ 10-ական համակարգում է, միայն պետք է հաշվի առնել, որ p-ական համակարգում տասի դերը կատարում է p-ն: Այժմ լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Բազմապատկել թվերը և ստուգելու համար կատարել նույն բազմապատկումը 10-ական համակարգում՝ 101101_2 և 101_2:

Լուծում՝ նախ թվերը բազմապատկենք 2-ական համակարգում՝

$$
\begin{array}{rl}
 & 101101_2 \\
* & \underline{\quad 101_2} \\
 & 101101 \\
+ & 000000 \\
 & \underline{101101\quad} \\
 & 11100001_2
\end{array}
$$

Այնուհետև թվերը բազմապատկենք 10-ական համակարգում՝

$$101101_2 = 45_{10},\ 101_2 = 5_{10},\ 45_{10} * 5_{10} = 225_{10}$$

Հեշտ է համոզվել, որ 11100001_2-ը 225_{10}-ի ներկայացումն է երկուական համակարգում:

Անցումը մի համակարգից մյուսին

Ծրագրավորման լեզուներում և ծրագրային ապահովման համակարգերում թվերի գրության մեջ օգտագործվում է տասնորդական կետը: Այսուհետև տասնորդական ստորակետի փոխարեն կօգտագործենք տասնորդական կետը:

Թվերի ձևափոխումը p-ական համակարգից q-ական համակարգի անվանենք անցում p-ական համակարգից q-ական համակարգի և նշանակենք $p \to q$: Դիտարկենք անցումները 10, 2, 8, 16 հիմքերով համակարգերի միջև: $2 \to 10,\ 8 \to 10\quad 16 \to 10$ անցումները ներկայացված են վերևի օրինակներում՝ 2-ական, 8-ական, 16-ական թվերի գրության և դրանց 10-ական արժեքների հաշվման միջոցով: Ամբողջ թիվը p-ական համակարգից 10-ականի ներկայացնելու համար հարկավոր է հաշվել $a_k \cdot p^k + a_{k-1} \cdot p^{k-1} \ldots + a_1 \cdot p^1 + a_0 \cdot p^0$ արտահայտության արժեքը: p-ն հաջորդաբար փակագծերից դուրս բերելով, ստացվում է հետևյալ արտահայտությունը, որը հայտնի է Հորների սխեմա անվանումով՝

$$((\ldots((a_k p + a_{k-1})p + a_{k-2})p \ldots + a_1)p + a_0$$

Հորների սխեմայով արտահայտության արժեքը հաշվելու համար յուրաքանչյուր քայլին պետք է ընթացիկ արժեքը բազմապատկել p-ով և գումարել հերթական թվանշանը:

10-ական համակարգից այլ համակարգերի, մասնավորապես՝ 2-ականի, 8-ականի, 16-ականի անցնելու ալգորիթմները տարբեր են ամբողջ թվերի և կոտորակային (մեկից փոքր) թվերի համար: Դիտարկենք $10 \to p$ անցումը ամբողջ թվերի համար: Տրված T_{10} թիվը 10-ական համակարգում բաժանում ենք p-ի 10-ական արժեքի վրա և հիշում քանորդն ու մնացորդը: Մնացորդը p-ական համակարգի թվանշան է: Այնուհետև քանորդը նորից ենք բաժանում p-ի վրա և հիշում քանորդն ու մնացորդը: Այս քայլերը կրկնում ենք, մինչև քանորդը դառնա 0: Վերջավոր քայլերից հետո ալգորիթմը կավարտվի (p>1): Ստացված մնացորդները հակառակ հերթականությամբ գրելով p-ական համակարգում՝ կունենանք պահանջվող թիվը: Օրինակ՝ տասական 101 թիվը բերել 2-ական համակարգի.

101 : 2 = 50, մնացորդը՝ 1
50 : 2 = 25, մնացորդը՝ 0
25 : 2 = 12, մնացորդը՝ 1
12 : 2 = 6, մնացորդը՝ 0
6 : 2 = 3, մնացորդը՝ 0
3 : 2 = 1, մնացորդը՝ 1
1 : 2 = 0, մնացորդը՝ 1

Արդյունքում ստացանք՝ $101_{10} = 1100101_2$

Դիտարկենք $10 \to p$ անցումը կոտորակային թվերի համար: Տրված է կոտորակային T_{10} թիվը $(0 < T_{10} < 1)$, որն ունի m հատ տասնորդական թվանշան, այսինքն՝ ներկայացված է 10^{-m} ճշտությամբ: T_p-ի թվանշանների քանակը նշանակենք n-ով: 10^{-m} ճշտությունը p-ական համակարգում պահպանելու համար n-ը պետք է բավարարի հետևյալ պայմանին՝

$p^{-n} \leq 10^{-m}$, որտեղից կստացվի $n \geq m/\lg p$:

Պայմանին բավարարող մինիմալ ամբողջ n-ը հավասար է թվանշանների անհրաժեշտ քանակին: $10 \to p$ անցումը կատարելու համար T_{10} թիվը 10-ական համակարգում բազմապատկում ենք p-ի 10-ական արժեքով, ստացվածից առանձնացնում ամբողջ մասը և թողնում կոտորակային մասը: Ամբողջ մասը p-ական համակարգի հերթական թվանշանն է: Այնուհետև կոտորակային մասը նորից ենք բազմապատկում p-ով, առանձնացնում ենք ամբողջ մասը և թողնում կոտորակային մասը: Այս քայլերը կրկնում ենք n անգամ: Ստացված ամբողջ մասերը նույն հերթականությամբ գրելով p-ական համակարգում՝ կունենանք պահանջվող թիվը: Օրինակ՝ տասական 0.8125 թիվը ձևափոխել 2-ական համակարգի: Նախ գտնենք պահաջվող n-ը: m=4, p=2: $n \geq m/\lg p = 4/\lg 2 \approx 4/0.301 \approx 13.3$, հետևաբար n-ը ստացվեց 14: Օգտվելով վերևում բերված ալգորիթմից՝ կատարենք $10 \to 2$ անցումը՝

0.8125 * 2 = 1.625 ամբողջ մասը՝ 1, կոտորակային մասը՝ 0.625
0.625 * 2 = 1.25 ամբողջ մասը՝ 1, կոտորակային մասը՝ 0.25
0.25 * 2 = 0.5 ամբողջ մասը՝ 0, կոտորակային մասը՝ 0.5
0.5 * 2 = 1.0 ամբողջ մասը՝ 1, կոտորակային մասը՝ 0.0

Արդյունքում ստացանք՝ $0.8125_{10} = 0.1101_2$

Այս դեպքում պահաջվող մինիմալ 14 թվանշանի փոխարեն բավարար եղան 4-ը, քանի որ 4-րդ քայլում կոտորակային մասը դարձավ 0:

Անցումները 2-ական համակարգից 2-ի աստիճան հանդիսացող համակարգերին և հակառակը կարելի է կատարել ավելի պարզ ալգորիթմներով: Դիտարկենք $2 \to 8$, $2 \to 16, 16 \to 2$ և $8 \to 2$ անցումները: Քանի որ $8 = 2^3$ և $16 = 2^4$, ապա կա փոխմիարժեք համապատասխանություն բոլոր 8-ական (16-ական) թվանշանների և բոլոր 2-ական եռյակների (քառյակների) միջև,

որոնք այդ թվանշանների 2-ական ներկայացումներն են: Այսպիսով՝ $8 \to 2$ և $16 \to 2$ անցնելիս պետք է թվի 8-ական (16-ական) թվանշանները փոխարինել համապատասխան եռյակներով (քառյակներով), իսկ $2 \to 8$ և $2 \to 16$ անցնելիս պետք է տասնորդական կետից սկսած, անջատել 2-ական եռյակներ (քառյակներ), լրացնել պակասող 0-ներով և փոխարինել դրանք համապատասխան 8-ական (16-ական) թվանշաններով: Օրինակներ՝

- $207.34_8 = 010\ 000\ 111\ .\ 011\ 100_2 = 10000111.0111_2$
- $3D4.B2_{16} = 0011\ 1101\ 0100\ .\ 1011\ 0010_2 = 1111010100.1011001_2$
- $11001110\ .\ 01101_2 = \mathbf{0}11\ 001\ 110\ .\ 011\ 01\mathbf{0}_2 = 316.32_8$
- $1011110110\ .\ 101001_2 = \mathbf{00}10\ 1111\ 0110\ .\ 1010\ 01\mathbf{00}_2 = 2F6.A4_{16}$

Ամբողջ թվերի կոդավորումը ԷՀՄ-ում

Ամբողջ թվերի ներկայացման համար օգտագործվում են կոդավորման երեք հիմնական եղանակներ՝ ուղիղ, հակադարձ և լրացուցիչ, որոնցից միայն վերջինն է լայնորեն օգտագործվում: Այդ կոդերը գրվում են 2-ական համակարգի թվանշանների միջոցով: Ենթադրենք՝ թվի ներկայացման համար հատկացված է n բիթանոց դաշտ: Այն սովորաբար զբաղեցնում է ամբողջ թվով բայթեր, օրինակ՝ $n = 8, 16, 32, 64$: Ամբողջ թվի ուղիղ կոդում 1-ին բիթը հատկացվում է թվի նշանի, իսկ մյուս $n-1$ բիթերը՝ թվի բացարձակ արժեքի գրառման համար: Ոչ բացասական թվի նշանի տեղում գրվում է 0, իսկ ոչ դրական թվի նշանի տեղում՝ 1: n բիթում կարելի է գրել 2^n հատ 2-ական n-յակներ: Դրանք ներկայացնում են $[-(2^{n-1}-1), 2^{n-1}-1]$ միջակայքի ամբողջ թվերի ուղիղ կոդերը: Ուղիղ կոդում կա երկու հատ 0՝ +0 և −0: Դիտարկենք օրինակներ $n = 8$ դեպքում՝

տասական թիվ	երկուական թիվ	ուղիղ կոդ
+0	+0	00000000
−0	−0	10000000
+10	+1010	00001010
−10	−1010	10001010
+35	+100011	00100011
−35	−100011	10100011
+127	+1111111	01111111
−127	−1111111	11111111

Հակադարձ կոդի սահմանումը հետևյալն է՝

$$[X]_{\text{հակ}} = \begin{cases} \text{X, եթե X} \geq 0 \\ 2^{\text{n}} - 1 - |\text{X}|, \text{եթե X} \leq 0 \end{cases}$$

Հակադարձ կոդով n բիթում ներկայացվում են $[-(2^{n-1} - 1), 2^{n-1} - 1]$ միջակայքի ամբողջ թվերը և կա երկու հատ 0՝ +0 և −0: Օրինակներ n = 8 դեպքում՝

տասական թիվ	երկուական թիվ	հակադարձ կոդ
+0	+0	00000000
−0	−0	11111111
+10	+1010	00001010
−10	−1010	11110101
+35	+100011	00100011
−35	−100011	11011100
+127	+1111111	01111111
−127	−1111111	10000000

Լրացուցիչ կոդը սահմանվում է հետևյալ կերպ՝

$$[X]_{\text{լր}} = \begin{cases} \text{X, եթե X} \geq 0 \\ 2^{\text{n}} - |\text{X}|, \text{եթե X} < 0 \end{cases}$$

Լրացուցիչ կոդով n բիթում ներկայացվում են $[-2^{n-1},\ 2^{n-1}-1]$ միջակայքի ամբողջ թվերը և կա մեկ հատ 0, որը բացի միարժեքությունից թույլ է տալիս կոդավորել նաև -2^{n-1} թիվը: Բացասական թվի լրացուցիչ կոդը կարելի է ստանալ հետևյալ կերպ՝

ա) ստանալ թվի ուղիղ կոդը,

բ) շրջել ուղիղ կոդի բիթերը (բացի նշանի բիթից)՝ ստանալով հակադարձ կոդը,

գ) ստացվածին գումարել մեկ:

Օրինակներ n = 8 բիթի դեպքում՝

տասական թիվ	երկուական թիվ	հակադարձ կոդ	լրացուցիչ կոդ
+0	+0	00000000	00000000
−0	−0	11111111	00000000
+10	+1010	00001010	00001010
−10	−1010	11110101	11110110
+35	+100011	00100011	00100011
−35	−100011	11011100	11011101
+127	+1111111	01111111	01111111
−127	−1111111	10000000	10000001
−128	−10000000	−	10000000

Որպես հակադարձ և լրացուցիչ կոդերով ներկայացված թվերի հետ թվաբանական գործողությունների կատարման օրինակ լուծենք հետևյալ խնդիրը՝

Օրինակ 2. Գտնել հետևյալ տարբերությունը՝ դրանց հակադարձ (լրացուցիչ) կոդերի (8 բիթում) գումարման միջոցով՝ 2 − 9: Նշել՝ որ դեպքում կառաջանա գերհագեցում:

Լուծում՝ նախ գրենք 2 և −9 թվերի հակադարձ (լրացուցիչ) կոդերը 8 բիթում. 2-ի հակադարձ և լրացուցիչ կոդը կլինի 00000010, իսկ −9-ի հակադարձ (լրացուցիչ) կոդը կլինի

11110110 (11110111), քանի որ 9-ի ուղիղ կոդը հավասար է 00001001:

Այժմ գումարենք իրար 2 և −9 թվերի հակադարձ (լրացուցիչ) կոդերը՝

$$\begin{array}{r} 00000010 \\ +\ 11110110 \\ \hline 11111000 \end{array} \qquad \begin{array}{r} 00000010 \\ +\ 11110111 \\ \hline 11111001 \end{array}$$

Այսպիսով, հակադարձ կոդերը գումարելիս ստացանք 11111000, որը −7-ի հակադարձ կոդն է 8 բիթում, իսկ լրացուցիչ կոդերը գումարելիս ստացանք 11111001, որը −7-ի լրացուցիչ կոդն է 8 բիթում: Երկու դեպքում էլ գերհագեցում չի առաջանում:

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն:

1. Ներկայացնել առաջին 20 բնական թվերը 10-ական, 2-ական, 3-ական, 5-ական և 8-ական համակարգերում:
2. Գրել նշված թվերին հաջորդող ամբողջ թվերը.

ա) 1_2	է) 1_8	լ) F_{16}
բ) 101_2	ը) 7_8	խ) $1F_{16}$
գ) 111_2	թ) 37_8	ծ) FF_{16}
դ) 1111_2	ժ) 177_8	կ) $9A9F_{16}$
ե) 101011_2	ի) 7777_8	հ) $CDEF_{16}$

3. Գրել նշված թվերին հաջորդող ամբողջ թվերը.

ա) 10_2	է) 10_8	լ) 10_{16}
բ) 1010_2	ը) 20_8	խ) 20_{16}
գ) 1000_2	թ) 100_8	ծ) 100_{16}
դ) 10000_2	ժ) 110_8	կ) $A10_{16}$
ե) 10100_2	ի) 1000_8	հ) 1000_{16}

4. Ի՞նչ թվանշանով կարող է ավարտվել.
   ա) 2-ական զույգ թիվը
   բ) 2-ական կենտ թիվը

գ) 3-ական զույգ թիվը

5. Ո՞ր ամենամեծ 10-ական թիվը կարելի է ներկայացնել երեք հատ թվանշաններով.
   ա) 2-ական համակարգում
   բ) 8-ական համակարգում
   գ) 16-ական համակարգում
6. Թվարկության ո՞ր համակարգում է ճշմարիտ հետևյալը.
   ա) `21 + 24 = 100`
   բ) `31 + 25 = 100`
   գ) `22 + 44 = 110`
7. p-ի ո՞ր արժեքի դեպքում է ճշմարիտ հետևյալը.
   ա) 59_{10} = 214p
   բ) 23_{10} = 32p
   գ) 100_{10} = 154p
8. Նշված թվերը բերել 10-ական համակարգի և ստուգելու համար կատարել հակառակ անցումները.

ա) 1011011_2	է) 517_8	լ) $1F_{16}$
բ) 10110111_2	ը) 1010_8	խ) ABC_{16}
գ) 011100001_2	թ) 1234_8	ծ) 1010_{16}
դ) 0.1000110_2	ժ) 0.34_8	կ) $0.A4_{16}$
ե) 110100.11_2	ի) 123.41_8	հ) $1DE.C8_{16}$

9. Տասական համակարգի թվերը բերել 2-ական, 8-ական և 16-ական համակարգերի: Ստուգելու համար կատարել հակառակ անցումները.
   ա) 125 բ) 229 գ) 88 դ) 37.25 ե) 206.125
10. Երկուական համակարգի թվերը բերել 8-ական և 16-ական համակարգերի և ստուգելու համար կատարել հակառակ անցումները.
    ա) 1001111110111.0111_2 դ) 1011110011100.11_2
    բ) 1110101011.1011101_2 ե) 10111.1111101111_2

գ) 10111001.101100111_2 զ) 1100010101.11001_2

11. Տասնվեցական համակարգի թվերը բերել 8-ական և 2-ական համակարգերի.
    ա) 2CE բ) 9F40 գ) ABCDE դ) 1010.101 ե) 1ABC.9D
12. Գրել ամբողջ թվերը.
    ա) 101101_2-ից մինչև 110000_2-ը 2-ական համակարգում
    բ) 202_3-ից մինչև 1000_3-ը 3-ական համակարգում
    գ) 14_8-ից մինչև 20_8-ը 8-ական համակարգում
    դ) 28_{16}-ից մինչև 30_{16}-ը 16-ական համակարգում
13. 10-ական համակարգի 47 և 79 թվերի համար գրել մի համակարգից մյուսին անցման հետևյալ շղթան.
    ա) 10 -> 2 -> 8 -> 10
    բ) 10 -> 8 -> 2 -> 16 -> 10
    գ) 10 -> 16 ->2 ->10
14. Կազմել միանիշ թվերի գումարման աղյուսակներ 3-ական և 5-ական համակարգերում:
15. Կազմել միանիշ թվերի բազմապատկման աղյուսակներ 3-ական և 5-ական համակարգերում:
16. Գումարել թվերը և ստուգելու համար կատարել գումարումը 10-ական համակարգում:
    ա) 1011101_2 և 1110111_2
    բ) 1011.101_2 և 101.011_2
    գ) 1011_2, 11_2 և 111.1_2
    դ) 1011_2, 11.1_2 և 111_2
    ե) 37_8 և 75_8
    զ) 165_8 և 37_8
    է) 7.5_8 և 14.6_8
    ը) 6_8, 17_8 և 7_8
    թ) A_{16} և F_{16}
    ժ) 19_{16} և C_{16}
    ի) $A.B_{16}$ և $E.F_{16}$
    լ) E_{16}, 9_{16} և F_{16}
17. Թվարկության ո՞ր համակարգերում են կատարված հետևյալ գումարումները: Գտնել յուրաքանչյուր համակարգի հիմքը:

ա) 98 + 89 = 121
բ) 1345 + 2178 = 3523
գ) 10101 + 1111 + 1011 = 20000
դ) 765 + 576 + 677 = 2462
ե) 98 + 56 + 79 = 167

18. Կատարել հանում.
ա) 10100_2-ից 111_2
բ) 10.11_2-ից 100.1_2
գ) 10010_2-ից 111.1_2
դ) 1110.11_2-ից 1011_2
ե) 20_8-ից 15_8
զ) 102_8-ից 47_8
է) 101_8-ից 56.7_8
ը) 30.01_8-ից 16.54_8
թ) 31_{16}-ից $1A_{16}$
ժ) $2A30_{16}$-ից $F9E_{16}$
ի) $B.92_{16}$-ից $D.1_{16}$
լ) 5678_{16}-ից ABC_{16}

19. Բազմապատկել թվերը և ստուգելու համար կատարել նույն բազմապատկումը 10-ական համակարգում:
ա) 101111_2 և 11_2
բ) 1111.01_2 և 11.01_2
գ) 1011.11_2 և 101.1_2
դ) 37_8 և 4_8
ե) 16_8 և 7_8
զ) 7.5_8 և 1.6_8
է) 101_2 և 1111.001_2
ը) 6.25_8 և 7.12_8

20. Բաժանել 1001110_2-ը 1010_2-ի և ստուգելու համար բազմապատկել քանորդն ու մնացորդը:

21. Բաժանել 10011010100_2-ը 1100_2-ի, այնուհետև կատարել բաժանումը 10-ական և 8-ական համակարգերում:

22. Հաշվել արտահայտությունների արժեքները.
ա) $256_8 + 10110.1_2 \cdot (60_8 + 12_{10}) - 1F_{16}$
բ) $1AD_{16} - 100101100_2 : 1010_2 + 217_8$
գ) $1010_{10} + (106_{16} - 11011101_2) \cdot 12_8$
դ) $1011_2 \cdot 1100_2 : 14_8 + (100000_2 - 40_8)$

23. Հետևյալ թվերը դասավորել աճման կարգով.
ա) 74_8, 110010_2, 70_{10}, 30_{16}
բ) $6E_{16}$, 142_8 1101001_2 , 100_{10}

գ) 777_8, 101111111_2, $2FF_{16}$, 500_{10}

դ) 100_{10}, 1100000_2, 60_{16}, 141_8

24. Գրել 3, 2, 1, 0, -1, -2, -3 թվերը մեկ բայթում (n=8) ուղիղ, հակադարձ և լրացուցիչ կոդերով:
25. Հետևյալ թվերը գրել ուղիղ կոդով (n=8 բիթ).
    31, -63, 65, -128
26. Հետևյալ թվերը գրել հակադարձ և լրացուցիչ կոդերով (n=8 բիթ).
    -9, -15, -127, -128
27. Գտնել լրացուցիչ կոդով գրված թվերի 10-ական ներկայացումը (n=8 բիթ).
    ա) 11111000 գ) 11101001
    բ) 10011011 դ) 10000000
28. Գտնել հակադարձ կոդով գրված թվերի 10-ական ներկայացումը (n=8 բիթ).
    ա) 11101000 բ) 10011111 գ) 10101011 դ) 10000000
29. Գտնել հետևյալ թվերի տարբերությունը՝ դրանց հակադարձ (լրացուցիչ) կոդերի (8 բիթում) գումարման միջոցով: Նշել՝ ո՞ր դեպքում կառաջանա գերհագեցում.
    ա) 9 – 2 դ) -20 - 10 է) -120 - 15
    բ) -2 – 9 ե) 50 - 25 ը) -126 - 1
    գ) -5 – 7 զ) 127 - 1 թ) -127 - 1

2. Փոփոխականներ, գործողություններ, արտահայտություններ

C++ լեզվի այբուբենին պատկանում են ոչ միայն լատիներեն տառերը, այլև բոլոր այն սիմվոլները, որոնք կարող են օգտագործվել ծրագրերում, այդ թվում նաև չերևացող սիմվոլները՝ բացատանիշը, տաբուլյացիան, նոր տողի սկիզբը։

Սիմվոլներից կազմվում են լեզվական մասնիկները կամ լեքսեմները։ Դրանք են՝ իդենտիֆիկատորները (նույնարկիչները), ծառայողական բառերը, հաստատունները կամ լիտերալները, գործողությունների նշանները և բաժանիչները։

Իդենտիֆիկատորները նախատեսված են ծրագրում օգտագործվող տարբեր միջոցներին անուն տալու համար։ Նրանք կարող են բաղկացած լինել տառերից, թվանշաններից և ընդգծման գծիկից։ Առաջին սիմվոլը պետք է լինի տառ կամ ընդգծման գծիկ։ Կոմպիլյատորը մեծատառերը և փոքրատառերը տարբերում է։ Այսինքն, a-ն և A-ն տարբեր իդենտիֆիկատորներ են։ Լեզվում իդենտիֆիկատորի երկարության վրա սահմանափակում չի դրվում, բայց կոմպիլյատորները տարբերիչ են համարում միայն առաջին սահմանափակ քանակով (հիմնականում 63) սիմվոլներ։

Ծառայողական բառերը հատուկ իմաստ ունեն և չեն կարող օգտագործվել որպես իդենտիֆիկատորներ։ Դրանց ցուցակը կարելի է գտնել տեղեկագրքերում և համացանցում։

Լիտերալները ծրագրում օգտագործվող հաստատուն աժեքներն են՝ ամբողջ և իրական թվերը, սիմվոլները (`'a'`, `'1'`), տողերը (`"Hello"`), և տրամաբանական արժեքները (`true`, `false`)։ Ամբողջ թվերը կարող են գրվել ոչ միայն տասական, այլև ութական (`023`) և տասնվեցական (`0xa1`) համակարգերում։ Իրական հաստատուններում կոտորակային մասն ամբողջ մասից անջատվում է կետով։ Կա նաև երկրորդ գրելաձև, որտեղ

օգտագործվում է `e` կամ `E` տառը, որը ցույց է տալիս աստիճանը։ Օրինակ՝ `314e-2`, նշանակում է `314`-ը բազմապատկած `10`-ի մինուս `2` աստիճանով, այսինքն՝ դա `3.14`-ն է։ Թվային հաստատունների վերջում կարող են լինել որոշակի պայմանանշաններ, որոնք կարող են փոխել հատկացվող հիշողության ծավալը կամ ձևաչափը։

Փոփոխականներն օգտագործվում են տարբեր տիպի տվյալներ պահելու համար։ Անվան միջոցով կարելի է փոխել պահվող արժեքը։ Այսպիսով, փոփոխականն ունի անուն, որ իդենտիֆիկատորն է և անվանում է հիշողության այն դաշտը, որտեղ ներկայացված է համապատասխան տվյալը, և ունի արժեք՝ այդ տվյալի արժեքը, որը կարող է փոփոխվել ծրագրի կատարման ընթացքում։

Ցանկացած փոփոխական ծրագրում օգտագործելուց առաջ պետք է նկարագրվի։ Նկարագրության մեջ նշվում է փոփոխականի տիպը։ Օրինակ՝ `int a;` Տիպի միջոցով սահմանվում է, թե տվյալ փոփոխականը ինչ արժեքներ կարող է ընդունել, և այդ փոփոխականի հետ ինչ գործողություններ կարելի է կատարել։ Բազային կամ տարրական տիպերից են `int`, `short`, `long`, `long long`, `enum`, `bool`, `char`, `float`, `double`, `long double` տիպերը։ Այս տիպի փոփոխականները ունենում են մեկ արժեք։ Բաղադրյալ տիպի փոփոխականները կարող են ունենալ մեկից ավել արժեքներ։ Նաև կարելի է սահմանել փոփոխական, որի արժեքը հանդիսանա այլ փոփոխականի կամ օբյեկտի հասցե։

Փոփոխականների սահմանումներում սկզբում գրվում է տիպը, ապա այդ տիպի փոփոխականների ցուցակը, որոնց ցանկության դեպքում կարելի է տալ սկզբնական արժեքներ։ Օրինակներ՝

```
int x=5; // նկարագրվել է x անունով ամբողջ տիպի
         // փոփոխականը, որի սկզբնական արժեքն է 5:
float f; // նկարագրվել է f անունով իրական float տիպի
         // փոփոխականը, որը սկզբնական արժեք չունի:
double d(3.1); // նկարագրվել է d անունով իրական double
               // տիպի փոփոխականը, որի սկզբնական
               // արժեքն է 3.1:
```

Կրկնակի թեք գծից հետո կարելի է գրել մեկնաբանություն: Այն ավարտվում է տվյալ տողի վերջում: Կարելի է գրել նաև մի քանի տողանոց մեկնաբանություն: Այդ դեպքում այն պետք է սկսվի `/*` սիմվոլներով և ավարտվի `*/` սիմվոլների զույգով:

C++-ում ստանդարտ մուտքային սարքից (ստեղնաշարից) տվյալների ներածման և ստանդարտ ելքային սարք (մոնիտոր) տվյալների արտածման համար նախատեսված են `cin` և `cout` գրադարանային միջոցները: Հետևյալ ծրագիրը ներածում է երկու ամբողջ թվեր և արտածում է նրանց գումարը՝

```
#include <iostream>
using namespace std;

int main()
{
    int a, b;
    cin>>a>>b;
    cout<<a+b<<endl;
    return 0;
}
```

`iostream`-ը ներածման և արտածման միջոցների գրադարանի անունն է: Երկրորդ տողում նշված է, որ ցանկանում ենք օգտագործել ստանդարտ անունների տիրույթում հայտարարված բոլոր գրադարանային միջոցները: Ծրագիրն ընդհանուր դեպքում բաղկացած է ֆունկցիաներից (ենթածրագրերից): Ճիշտ մեկ

ֆունկցիայի անունը պետք է լինի `main`: Ծրագրի աշխատանքը սկսվում է `main`-ից: Ծրագրի կատարման վերջում `main`-ը `return` հրամանի միջոցով օպերացիոն համակարգին փոխանցում է `0` արժեքը, որը նշանակում է, որ ծրագիրը հաջողությամբ ավարտվել է:

Գործողություններ

`C++` լեզվում սահմանված են գործողություններ, որոնք կարելի է կատարել որոշակի ներկառուցված տիպի մեծությունների հետ: Գործողությանը մասնակցող մեծությունը անվանում ենք օպերանդ: Կախված նրանից, թե քանի օպերանդ է մասնակցում տվյալ գործողությունը կատարելու ժամանակ, գործողությունները կարող են լինել ունար (1 օպերանդ), բինար (2 օպերանդ) և տերնար (3 օպերանդ): Գործողությունները օպերանդների հետ կազմում են արտահայտություններ: Գործողության նշանները կարող են բաղկացած լինել մեկ, երկու կամ երեք նիշերից: Մեկից ավելի նիշերի հաջորդականությունը դիտարկվում է որպես մեկ ամբողջություն: Հետևյալ աղյուսակում բերված են `C++` լեզվի հիմնական գործողությունները, որտեղ նշված են գործողության նախապատվությունը և ասոցիատիվությունը: Նախապատվությունը որոշում է արտահայտությունում տարբեր գործողությունների առկայության դեպքում նրանց կատարման հերթականությունը: Ասոցիատիվությունը ցույց է տալիս, թե գործողությունների միևնույն նախապատվությունների դեպքում առաջինը որ գործողությունը պետք է կատարվի՝ աջակողմյանը, թե ձախակողմյանը: Աղյուսակում գործողությունները ներկայացված են նախապատվությունների նվազման կարգով՝

Նախապատ-վություն	Գործողու-թյուն	Նկարագրություն	Ասոցիատի-վություն
1	::	Տեսանելիության տի-րույթ	Ձախից աջ
2	a++ a--	Ետադիր ինկրեմենտ և դեկրեմենտ	
	()	Ֆունկցիայի կանչ	
	[]	Դիմում զանգվածի տարրին ինդեքսի միջոցով	
	.	Տարրի ընտրություն	
	–>	Ցուցիչով տարրի ընտրություն	
3	++a –a	Նախադիր ինկրե-մենտ և դեկրեմենտ	Աջից ձախ
	+ –	Ունար պլյուս և մի-նուս	
	! ~	Տրամաբանական NOT Բիթային NOT	
	(*type*)	Տիպի բերումը type տիպի	
	*	Հասցեի միջոցով օ-պերանդի արժեքին դիմելու գործողու-թյուն	

	&	Օպերանդի հասցե	
	sizeof	Չափը բայթերով	
	new, new[]	Դինամիկ հիշողության առանձնացում	
	delete, delete[]	Դինամիկ հիշողության ազատում	
4	.* ->*	Ցուցիչ անդամի վրա	Ձախից աջ
5	* / %	Բազմապատկում, բաժանում և մնացորդ	
6	+ −	Գումարում և հանում	
7	<< >>	Բիթային ձախ և աջ տեղաշարժ	
8	< <=	< և ≤	
	> >=	> և ≥ համեմատության գործողություններ	
9	== !=	= և ≠ համեմատության գործողություններ	
10	&	Բիթային AND	
11	^	Բիթային XOR	
12	|	Բիթային OR	
13	&&	Տրամաբանական	

		AND	
14	||	Տրամաբանական OR	
15	?:	Պայմանի տերնար գործողություն	Աջից ձախ
	=	Վերագրման գործողություն	
	+= −=	Գումարումով և հանումով վերագրումներ	
	*= /= %=	Բազմապատկումով, բաժանումով և մնացորդով վերագրումներ	
	<<= >>=	Բիթային ձախ և աջ տեղաշարժումներով վերագրումներ	
	&= ^= |=	Բիթային (AND, XOR, OR) գործողություններով վերագրումներ	
16	throw	Բացառման դուրս բերման գործողություն	
17	,	Ստորակետ	Ձախից աջ

Դիտարկենք այդ գործողություններից մի քանիսը։

Թվաբանական գործողություններ

`+, -, *, /, %, ++, --`

`/` բաժանման գործողությունը կարող է կիրառվել սահող ստորակետով և ամբողջ թվերի համար: Եթե երկու օպերանդներն էլ ամբողջ են, ապա արդյունքում ստացվում է ամբողջ թիվ (կլորացումը տեղի է ունենում դեպի զրոյի կողմը): Օրինակ `-5/2`-ի արժեքը `-2` է: Եթե օպերանդներից գոնե մեկը սահող ստորակետով տիպի է, ապա արդյունքում ստացվում է սահող ստորակետով թիվ՝ բաժանման արժեքը որոշակի ճշտությամբ:

`%` գործողության օպերանդներն ամբողջ տիպի են, արդյունքը՝ բաժանման մնացորդն է: Օրինակ՝`11%3`-ի արժեքը `2` է, `7%(-3)`-ի արժեքը `1` է, իսկ `-7%3`-ի արժեքը `-1` է: Մնացորդի նշանը համընկնում է բաժանելիի նշանի հետ:

`++` գործողությունը (ինկրեմենտ) ավելացնում է օպերանդի արժեքը մեկով, իսկ `--` գործողությունը (դեկրեմենտ) պակասեցնում է օպերանդի արժեքը մեկով: Այս գործողությունների օպերանդը փոփոխական է, որի տիպը կարող է լինել ամբողջ (`char, short, int, long`) կամ ցուցչային: Ունեն օգտագործման երկու ձև. նախադիր՝ երբ գործողության նշանը դրվում է օպերանդից առաջ և ետադիր՝ երբ գործողության նշանը դրվում է օպերանդից հետո: Նախադիրի օգտագործման դեպքում նախ փոփոխվում է փոփոխականի արժեքը մեկով, այնուհետև օգտագործվում է փոփոխականի նոր արժեքը: Ետադիրի օգտագործման դեպքում օգտագործվում է փոփոխականի արժեքը, որից հետո այն փոխվում է մեկով: `bool` տիպի օպերանդի հետ կարելի է կատարել միայն `++` գործողությունը, ինչպես նախադիր, այնպես էլ ետադիր ձևերով: Օրինակ՝

```
int a=3,b,c=3,d;
b=++a; //a-ի և b-ի արժեքները դառնում են 4
d=c++; //d-ի արժեքը դառնում է 3, c-ի արժեքը դառնում է 4
```

Համեմատության գործողություններ

`<, <=, ==, !=, >=, >`

Այս գործողությունների արդյունքը տրամաբանական տիպի արժեք է՝ `true` կամ `false`: Քանի որ C++ լեզվում գործում է անբացահայտ տիպի ձևափոխություն տրամաբանական տիպից դեպի ամբողջ տիպ (`true`-ն դառնում է `1`, իսկ `false`-ը՝ `0`), ապա համեմատման գործողության արդյունքը կարող է հանդիսանալ թվաբանական գործողության օպերանդ: Օրինակ՝ եթե `a=3` և `b=4`, ապա `(a<b)+6` արտահայտության արժեքը `7` է:

Տրամաբանական գործողություններ

`&&, ||, !`

Այս գործողությունների օպերանդները և արդյունքը `bool` տիպի են: `&&` (տրամաբանական և) գործողության արդյունքը true է, եթե օպերանդները երկուսն էլ `true` են, հակառակ դեպքում արդյունքը `false` է: || (տրամաբանական կամ) գործողության արդյունքը `true` է, եթե օպերանդներից գոնե մեկը `true` է, հակառակ դեպքում արդյունքը `false` է: `!` (տրամաբանական ոչ) գործողությունը ունար գործողություն է, որը ժխտում է օպերանդի արժեքը: Այսինքն, եթե օպերանդի արժեքը `true` է, ապա գործողության արդյունքը `false` է, իսկ եթե օպերանդի արժեքը `false` է, ապա գործողության արդյունքը `true` է:

Բիթային գործողություններ

`&, |, ^, ~, <<, >>`

Այս գործողությունների օպերանդներն ամբողջ տիպի են, և գործողությունները կատարվում են նրանց կոդերի հետ: `&,|,^` բինար գործողությունները կիրառվում են օպերանդների նույն կարգում գտնվող բիթերի զույգերի հետ: Այդ գործողությունները զույգ բիթերի համար որոշվում են հետևյալ կերպ՝

- `&` - բիթ առ բիթ բազմապատկում
  `0 & 0 = 0 & 1 = 1 & 0 = 0; 1 & 1 =1;`
- `|` - բիթ առ բիթ գումարում
  `1 | 1 = 0 | 1 = 1 | 0 = 1;  0 | 0 = 0;`
- `^` - բացառող կամ (XOR)
  (արդյունքը հավասար է `0`, եթե բիթերը հավասար են, հակառակ դեպքում՝ `1`)
  `0 ^ 0 = 1 ^ 1 =0; 1 ^ 0 = 0 ^ 1 = 1;`

Օրինակ՝

```
char c1='5',c2='6',c3; //'5'-ի կոդն է 00110101
                       //'6'-ի կոդն է 00110110
c3=c1&c2;              //c3-ի կոդն է 00110100, որը
                       //'4'–ի կոդն է
cout<<c3<<endl;        //կտպվի 4
c3=c1|c2;              //c3–ի կոդն է 00110111, որը
                       //'7'–ի կոդն է
cout<<c3<<endl;        //կտպվի 7
c1='0',c2='A';         //'A'-ի կոդն է 01000001
                       //'0'-ի կոդն է 00110000
C3=c1^c2;              //c3-ի կոդն է 10001110, որը
                       //'q'-ի կոդն է
cout<<c3<<endl;        //կտպվի q
```

Բիթային ժխտումը (~) ունար գործողություն է, որը փոխում է օպերանդի յուրաքանչյուր բիթի արժեքը՝ `1`-ը դարձնում է `0`, իսկ `0`-ն դարձնում է `1`: Տեղաշարժի (<< և >>) գործողությունները առաջին օպերանդի երկուական կոդը տեղաշարժում են ձախ և աջ երկրորդ օպերանդում նշված բիթերի քանակով: Ձախ տեղաշարժի ժամանակ աջից ազատված բիթերը լրացվում են `0`-ներով: Աջ տեղաշարժի ժամանակ ձախից ազատված բիթերը լրացվում են

0-ներով, եթե առաջին օպերանդը unsigned տիպի է և նշանի բիթով, եթե առաջին օպերանդը նշանով տիպի է:

Վերագրման գործողություններ

= վերագրման ձախ կողմում պետք է լինի փոփոխական (left-value), իսկ աջ կողմում՝ արտահայտություն (right-value): Հաշվվում է արտահայտության արժեքը, որը վերագրվում է փոփոխականին, այսինքն՝ արտահայտության արժեքը գրանցվում է փոփոխականին հատկացված հիշողության դաշտում: Այդ դաշտի նախկին արժեքը ջնջվում է: Օրինակ՝

```
double d(2.3),z=5.1;
cout<<"z="<<z<<endl; //կտպվի z=5.1
cout<<"d="<<d<<endl; //կտպվի d=2.3
z=z+d;
cout<<"z="<<z<<endl; //կտպվի z=7.4
```

θ= վերագրման գործողությունների (**θ** ∈ {*, /, %, +, −, ≪ ,≫, &, |, ^}) ձախ կողմում փոփոխական է (left-value), իսկ աջ կողմում՝ արտահայտություն՝

փոփոխական **θ**= *արտահայտություն*

Այն համարժեք է հետևյալ վերագրման գործողությանը՝

փոփոխական = փոփոխական **θ** *արտահայտություն*

Աջ և ձախ մասերում հանդես է գալիս նույն փոփոխականը: Ընդ որում՝ **θ**= գործողությունն իմաստ ունի, եթե **θ** գործողությունը որոշված է փոփոխականի տիպի համար: Օրինակ՝

```
int x=15,y=7;
x+=3*16/y;              //համարժեք է x=x+3*16/y
cout<<"x="<<x<<endl; //կտպվի x=21
```

Այլ գործողություններ

`sizeof` գործողությունն ունի օգտագործման երկու ձև՝

- `sizeof(`*տիպ*`)`
- `sizeof` *արտահայտություն*

Առաջին դեպքում `sizeof` գործողության արդյունքը տվյալ տիպի փոփոխականին հատկացվող հիշողության բայթերի քանակն է, երկրորդ դեպքում՝ արտահայտության արժեքի զբաղեցրած բայթերի քանակը: Օրինակ՝

```
double z=2,t=0.0;
cout<<sizeof z*t; //8, քանի որ z*t-ն double տիպի է
```

`?:` տերնար պայմանական գործողության ընդհանուր տեսքը հետևյալն է՝

արտահայտություն1 `?` *արտահայտություն2* `:` *արտահայտություն3*

Եթե *արտահայտություն1*-ի արժեքը `true` է, ապա գործողության արդյունքը հավասար է *արտահայտություն2*-ի արժեքին, հակառակ դեպքում գործողության արդյունքը հավասար է *արտահայտություն3*-ի արժեքին:

Այժմ լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Ներածել քառանիշ թիվ: Արտածել թվի թվանշանները՝ դասավորված հակառակ կարգով (օր. `1265`, `5621`):

Լուծում՝

```
#include <iostream>

using std::cin;
using std::cout;

int main()
{
    int n;
    cin>>n;
    short digit1=n/1000; //առաջին թվաշանը
```

```
    short digit2=n/100%10; //երկրորդ թվանշանը
    short digit3=n/10%10; //երրորդ թվանշանը
    short digit4=n%10; //վերջին թվանշանը
    cout<<digit4<<digit3<<digit2<<digit1;
    return 0;
}
```

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն:

1. Նշված գործողություններից առանձնացնել `left-value` հատկությամբ օժտված օպերանդ ունեցող գործողությունները.
   `+, ++, +=, %, ?:, &, &=, *, !, =, ==`
2. Նկարագրված փոփոխականների համար բերել դրանց ներքին կոդավորումը.
   ա) `int x=5; int y=-5;`
   բ) `float x=5; float y=-5;`
   գ) `double x,y;`
   դ) `char x;`
   ե) `short x;`
   զ) `long x;`
   է) `int x;`
   ը) `unsigned int x;`
3. Տրված է `x` փոփոխականի կոդը `16`-ական համակարգում՝ `0x7FFFFFF6`: Գտնել այդ փոփոխականի արժեքը, եթե `x`-ը հետևյալ տիպի է.
   ա) `int`
   բ) `float`
   գ) `signed int`
   դ) `unsigned int`

4. Տրված է `x` փոփոխականի կոդը `16`-ական համակարգում՝ `0x96`: Գտնել այդ փոփոխականի արժեքը, եթե `x`-ը հետևյալ տիպի է.
   ա) `bool`
   բ) `char`
   գ) `unsigned char`
   դ) `signed char`
5. Դասավորել նշված տիպերն ըստ աճման կարգի.
   `int, unsigned int, signed int, char, long int, short, float`:
6. Հաշվել արտահայտության արժեքը:
   ա) `5>6>4`
   բ) `6>5>4`
   գ) `6>5<4`
   դ) `6>=5<=4`
7. `x`-ի ի՞նչ արժեքների դեպքում արտահայտության արժեքը ճիշտ կլինի.
   ա) `x>7<10`
   բ) `7>x>4`
   գ) `x=5>12`
   դ) `x==8<7`
8. Ունենք `int x=5,y=2; float z=5.2f; char c='2';` փոփոխականները: Հաշվել արտահայտության արժեքը:
   ա) `x+y-z`
   բ) `x%y*z`
   գ) `z=x%y`
   դ) `z+x--/++y`
   ե) `z=x>>y`
   զ) `x&y|~x`
   է) `x|~x`

ը) `!x&x+y`

թ) `x-=y-2+3*x`

ժ) `z=2.0+c+x%y`

ի) `c='1'; c=c<<2; c>>=2`

լ) `++x-y>=5 && y`

9. Ի՞նչ կարտածի հետևյալ ծրագիրը.

ա)
```
#include <iostream>
using namespace std;
int main(){
    char c=0x5A;
    char d=0123;
    int e=90;
    cout<<(c*d-d*89)<<endl;
    cout<<oct<<(int)c<<endl;
    cout<<hex<<(int)d<<endl;
    cout<<hex<<(e<0?c-d:c+d)<<endl;
    cout<<dec<<(e%4)<<endl;
    return 0;
}
```

բ)
```
#include <iostream>
using namespace std;
int main(){
    char c=-128;
    unsigned char d=128;
    int e;
    e=d/c;
    cout<<oct<<e<<endl;
    cout<<oct<<(c^d)<<endl;
    cout<<dec<<(e*10+e)<<endl;
    cout<<hex<<~e<<endl;
    cout<<hex<<(e=d,e%3)<<endl;
    return 0;
}
```

10. Ի՞նչ կարտածի հետևյալ ծրագիրը.

ա)
```
#include <iostream>
using namespace std;
int main(){
    char c1='a',c2=011;
    int a=0xc;
    cout<<dec<<(c1+=a%c2--)<<' ';
    cout<<c2+a*2<<endl;
    cout<<oct<<(!(c1=='h' || a%4==3)
          ?a+4:--a)<<endl;
    cout<<hex<<(a/4+(c2/=2))*4<<endl;
    cout<<dec<<c1<<' '<<(int)c2<<' '<<a
        <<endl;
    return 0;
}
```

բ)
```
#include <iostream>
using namespace std;
int main(){
    char c1=100,c2=0x31;
    short s=0x1b;
    cout<<hex<<-(s+=++c1-c2--)<<endl;
    cout<<dec<<(c1%7==6?++s:s--)<<endl;
    cout<<oct<<(s*=2)-c1<<' '<<s<<endl;
    cout<<dec<<(int)c1<<' '<<(int)c2<<' '
        <<s<<endl;
    return 0;
}
```

11. Ի՞նչ կարտածի ծրագրի հետևյալ հատվածը.

ա)
```
int bitstring1=111; int bitstring2=-67;
cout<<bitstring1|~bitstring2;
cout<<~bitstring1|bitstring2;
```

բ)
```
int x=14, y=4;
y=y<<2;
cout<<++x;
cout<<y;
```

12. Ինչի՞ են հավասար `x,z,y1,y2` փոփոխականների արժեքները հետևյալ գործողությունները կատարելուց հետո.

ա)
```
int x,z,y1=5,y2=30;
y1=y1<<1;
x=--y1+20;
z=20+y2++;
```

բ)
```
int x,z,y1=20,y2=10;
x=++y1+20;
z=20+y2--;
```

գ)
```
int x,z,y1=5,y2=30;
y1=y1<<1;
x=--y1+20;
z=20+y2++;
```

13. Արտածե՛ք Ձեր անունն ու ազգանունը առանձին տողերում:
14. Արտածել հետևյալ պատկերները.

```
* * * *                 *               *
*     *               * * *             * *
*     *             * * * * *           * * *
* * * *               * * *             * * * *
                        *               * * * * *
```

15. Ներածել երկու ամբողջ թիվ և արտածել դրանց գումարը, տարբերությունն ու արտադրյալը:
16. Ներածել երկու բնական թիվ և արտածել դրանց միջին թվաբանականն ու միջին երկրաչափականը:
17. Տրված է խորանարդի կողմը: Ստանալ խորանարդի ծավալը և կողմնային մակերևույթի մակերեսը:
18. Տրված են ուղղանկյան երկու կողմերի երկարությունները: Գտնել այդ ուղղանկյան
    ա) պարագիծը,
    բ) մակերեսը,
    գ) անկյունագիծը:

19. Տրված են եռանկյան երեք կողմերի երկարությունները: Գտնել այդ եռանկյան
    ա) պարագիծը,
    բ) մակերեսը,
    գ) որևէ երկու կողմերի կազմած անկյան սինուսը,
    դ) որևէ երկու կողմերի կազմած անկյան կոսինուսը,
    ե) որևէ կողմին տարված բարձրությունը:
20. Ներածել ուղղանկյուն եռանկյան էջերը, գտնել և արտածել ներքնաձիգն ու եռանկյան մակերեսը:
21. Ներածել շրջանագծի տրամագիծը և արտածել նրա երկարությունը:
22. Ներածել քառանիշ թիվ: Արտածել թվի թվանշանները՝ դրանց միջև տեղադրելով գծիկներ (օր. 6508, 6-5-0-8):
23. Ներածել մի թիվ: Այն բազմապատկել 5-ով, գումարել 6, բազմապատկել 4-ով, գումարել 9, բազմապատկել 5-ով: Ջնջել ստացված թվի վերջին երկու թվանշանները, հանել 1 և արտածել:
24. Ներածել ծննդյան թիվ և տարիք: Տարեթիվը կրկնապատկել, գումարել 5, բազմապատկել 50-ով, գումարել տարիքը, հանել 250 և բաժանել 100-ի: Ստացված թիվն արտածել էկրանին ստորակետից հետո 2 նիշի ճշտությամբ:

3. Ճյուղավորում

Ճյուղավորում կազմակերպելու համար C++ լեզվում կարելի է օգտագործել պայմանի երկու կառույց՝ `if` / `if…else` և `switch`: `if` կառույցն ունի հետևյալ տեսքը՝

```
if(պայման)
   հրաման
```

պայման-ը տրամաբանական արտահայտություն է, *հրաման*-ը կատարվում է միայն այն դեպքում, երբ *պայման*-ի արժեքը `true` է: *հրաման*-ը C++-ի ցանկացած հրաման է և կարող է իրենից ներկայացնել նաև բաղադրյալ հրաման՝ ձևավոր փակագծերում առնված հրամանների համախմբություն: Օրինակ՝

```
if(k<7)
   k+=5;
```

Քանի որ C++ լեզվում գործում է անբացահայտ ձևափոխություն թվային տիպերից դեպի տրամաբանական տիպ (`0`-ն դառնում է `false`, իսկ մնացած թվերը՝ `true`), ապա *պայման*-ը կարող է լինել նաև թվաբանական արտահայտություն:

Դիտարկենք `if` կառույցի ընդլայնված տարբերակը՝ `if…else` կառույցը, որն ունի հետևյալ տեսքը՝

```
if(պայման)
   հրաման1
else
   հրաման2
```

հրաման1-ը կատարվում է միայն այն դեպքում, երբ *պայման*-ի արժեքը `true` է: Հակառակ դեպքում կատարվում է *հրաման2*-ը: Օրինակ՝

```
if(a<b)
  min=a;
else
  min=b;
```

Այժմ, օգտագործելով `if…else` կառույցը, լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Տրված են երեք ամբողջ թվեր: Արտածել մեծությամբ առաջին երկուսի գումարը:

Լուծում՝

```
#include <iostream>

using namespace std;

int main()
{
    int first, second, third;
    cin>>first>>second>>third;
    if(first>second)
        if(second>third)//ամենափոքրը երրորդն է
            cout<<first+second;
        else//ամենափոքրը երկրորդն է
            cout<<first+third;
    else
        if(third>first)//ամենափոքրը առաջինն է
            cout<<second+third;
        else //ամենափոքրը երրորդն է
            cout<<first+second;
    return 0;
}
```

Այժմ դիտարկենք պայմանական մյուս կառույցը՝ `switch`-ը: Ընդհանուր դեպքում այն ունի հետևյալ տեսքը՝

```
switch(արտահայտություն)
{
case հաստատուն1:
    հրաման1
…
```

```
case հաստատուն:
    հրաման
default:
    հրաման
}
```

արտահայտություն-ը այնպիսի արտահայտություն է, որի արժեքը ինտեգրալ տիպի է, *հաստատուն1,…,հաստատուն*-ը իրարից տարբեր հաստատուններ են, որոնց տիպը համընկնում է *արտահայտություն*-ի տիպի հետ, *հրաման1*-ը,…, *հրամանn*-ը, *հրաման*-ը հրամաններ են, $n \geq 0$: Ընդ որում՝ `default`-ը և նրանից հետո եկող *հրաման*-ը կարող են բացակայել: `switch` կառույցն աշխատում է հետևյալ կերպ՝

1. հաշվարկվում է *արտահայտություն*-ի արժեքը և հերթականությամբ համեմատվում է `case`-երի դիմացի հաստատունների հետ,
2. եթե $\nexists i \in \{1, \dots, n\}$ այնպիսին, որ առաջին քայլում հաշվարկված արժեքը հավասար է *հաստատուն_i*-ին, ապա կատարվում է *հրաման*-ը և `switch`-ի աշխատանքն ավարտվում է: Եթե այդ ընթացքում հանդիպում է `break` հրամանը, ապա կատարումն ընդհատվում է և `switch`-ի աշխատանքն ավարտվում է,
3. եթե $\exists i \in \{1, \dots, n\}$ այնպիսին, որ առաջին քայլում հաշվարկված արժեքը հավասար է *հաստատուն_i*-ին, ապա կատարվում է *հրաման_i-ը, հրաման_{i+1}-ը,…, հրամանn-ը և հրաման*-ը: Եթե այդ ընթացքում հանդիպում է `break` հրամանը, ապա կատարումն ընդհատվում է և `switch`-ի աշխատանքն ավարտվում է:

Սովորաբար `switch`-ի հրամաններից յուրաքանչյուրի վերջում հանդիպում է `break` հրամանը, որպեսզի որևէ համընկնումից և համապատասխան հրամանի կատարումից հետո հաջորդ

հրամանները ևս չկատարվեն: Այժմ, օգտագործելով `switch` կառույցը, լուծենք հետևյալ խնդիրը՝

Օրինակ 2. Տրված a և b ամբողջ թվերի և **θ** սիմվոլի համար հաշվել և արտածել a **θ** b արտահայտության արժեքը, որտեղ **θ**-ն կարող է լինել `+`, `-`, `*` կամ `/` գործողություններից որևէ մեկը:

Լուծում՝

```
#include <iostream>

using std::cin;
using std::cout;
using std::endl;

int main()
{
    int a, b;
    char c;
    cin>>a>>c>>b;
    switch(c)
    {
    case '+':
        cout<<a+b<<endl;
        break;
    case '-':
        cout<<a-b<<endl;
        break;
    case '*':
        cout<<a*b<<endl;
        break;
    case '/':
        cout<<a/b<<endl;
        break;
    default:
        cout<<"Wrong symbol was provided!\n";
    }
```

```
    return 0;
}
```

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն:

1. Տրված են a և b բնական թվերը: Արտածել YES, եթե
   ա) դրանք երկուսն էլ կենտ են,
   բ) դրանցից գոնե մեկը կենտ է,
   գ) դրանցից ճիշտ մեկը կենտ է,
   դ) դրանք երկուսն էլ կենտ են կամ երկուսն էլ զույգ:
   Հակառակ դեպքում արտածել NO:
2. Տրված են a, b և c ամբողջ թվերը: Արտածել YES, եթե դրանցից
   ա) գոնե մեկը զրո է,
   բ) ճիշտ մեկը հինգ է,
   գ) ոչ մեկը մյուսներին հավասար չէ,
   դ) ճիշտ երկուսն են հավասար,
   ե) բոլոր երեքն իրար հավասար են,
   զ) գոնե մեկը բացասական է,
   Հակառակ դեպքում արտածել NO:
3. Տրված են a, b և c ամբողջ թվերը: Արտածել YES, եթե
   ա) երեքն էլ դրական են,
   բ) դրանցից գոնե մեկը դրական է,
   գ) դրանցից ճիշտ մեկը դրական է,
   դ) դրանցից ճիշտ երկուսը դրական են,
   ե) դրանցից գոնե երկուսն իրար հավասար են,
   զ) դրանցից գոնե երկուսն իրարից միայն նշանով են տարբերվում:
   Հակառակ դեպքում արտածել NO:
4. Տրված է a դրական ամբողջ թիվը: Արտածել YES, եթե այն
   ա) երկնիշ և զույգ թիվ է,

բ) եռանիշ և կենտ թիվ է:

Հակառակ դեպքում արտածել `NO`:

5. Տրված է `a` եռանիշ թիվը: Արտածել `YES`, եթե
   ա) նրա բոլոր թվանշանները տարբեր են,
   բ) նրա թվանշանները կազմում են աճող հաջորդականություն,
   գ) այն սիմետրիկ (պոլինդրոմ) է, այսինքն աջից ձախ և ձախից աջ նույն կերպ է կարդացվում:

   Հակառակ դեպքում արտածել `NO`:
6. Տրված են `x,y` թվերը: Արտածել `YES`, եթե `(x,y)` կոորդինատներով կետն ընկած է կոորդինատային հարթության
   ա) երկրորդ քառորդում,
   բ) չորրորդ քառորդում,
   գ) երկրորդ կամ երրորդ քառորդում,
   դ) առաջին կամ երրորդ քառորդում:

   Հակառակ դեպքում արտածել `NO`:
7. Տրված են `x,y,x1,y1,x2,y2` ոչ բացասական ամբողջ թվերը: Արտածել `YES`, եթե `(x,y)` կոորդինատներով կետն ընկած է այն ուղղանկյան ներսում, որի ձախ վերևի անկյան կոորդինատներն են `(x1,y1)`, իսկ աջ ներքևի անկյանը՝ `(x2,y2)`, և որի կողմերը զուգահեռ են կոորդինատների առանցքներին: Հակառակ դեպքում արտածել `NO`:
8. Տրված են `a,b,c` բնական թվերը: Արտածել `YES`, եթե `a,b,c` կողմերով եռանկյուն գոյություն ունի, `NO`՝հակառակ դեպքում:
9. Տրված են `a,b,c` բնական թվերը, որոնք եռանկյան կողմերի երկարություններն են: Արտածել `YES`, եթե `a,b,c` կողմերով եռանկյունը
   ա) հավասարակողմ է,
   բ) հավասարասրուն է,

գ) ուղղանկյուն է:

Հակառակ դեպքում արտածել `NO`:

10. Տրված են `x,y` ամբողջ թվերը, որոնք շախմատի տախտակի մի վանդակի կոորդինատներն են (այսինքն այդ թվերն ընկած են `[1,8]` հատվածում): Հաշվի առնելով, որ `(1,1)` վանդակը սև է, արտածել `white`, եթե `(x,y)` վանդակը սպիտակ է: Հակառակ դեպքում արտածել `black`:
11. Տրված են `x1,y1` և `x2,y2` թվազույգերը, որոնք շախմատի տախտակի երկու տարբեր վանդակների կոորդինատներ են (այսինքն այդ թվերն ընկած են `[1,8]` հատվածում): Արտածել `YES`, եթե այդ վանդակները նույն գույնի են, `NO`՝հակառակ դեպքում:
12. Տրված են երեք ամբողջ թվեր: Արտածել
    ա) դրանցից փոքրագույնի արժեքը,
    բ) դրանցից մեծագույնի արժեքը
    գ) մեծությամբ երկրորդ թվի արժեքը,
13. Տրված է `(0,0)` կենտրոնով և `r` շառավղով շրջանագիծ: Արտածել `YES`, եթե տրված `(x,y)` կետը գտնվում է
    ա) շրջանագծի վրա,
    բ) շրջանի մեջ,
    գ) շրջանից դուրս:

    Հակառակ դեպքում արտածել `NO`:
14. Տրված է `(a,b)` կենտրոնով և `r` շառավղով շրջանագիծ: Արտածել `YES`, եթե տրված `(x,y)` կետը գտնվում է
    ա) շրջանագծի վրա,
    բ) շրջանի մեջ,
    գ) շրջանից դուրս:

    Հակառակ դեպքում արտածել `NO`:
15. Տրված են երկու եռանկյուններ՝ համապատասխան կողմերով: Արտածել `YES`, եթե դրանք

ա) հավասար են,

բ) նման են:

Հակառակ դեպքում արտածել `NO`:

16. Տրված են `a,b,c` իրական թվերը: Եթե դրանք դասավորված են աճման կամ նվազման կարգով, դրանց արժեքները կրկնապատկել, հակառակ դեպքում յուրաքանչյուրի նշանը փոխել: Արտածել ստացված թվերը:
17. Թվային առանցքի վրա տրված են `a,b,c` կետերը: Պարզել, թե `b` և `c` կետերից որն է ավելի մոտ `a`-ին, արտածել նրա արժեքը և հեռավորությունը `A` կետից:
18. Տրված են կետի կոորդինատները կոորդինատային հարթությունում: Եթե կետը համընկնում է կոորդինատների սկզբնակետի հետ՝ արտածել 0: Եթե չի համընկնում կոորդինատների սկզբնակետի հետ, բայց ընկած է `x`-երի կամ `y`-ների առանցքներից որևէ մեկի վրա՝ արտածել համապատասխանաբար՝ 1 կամ 2: Եթե կետը ընկած չէ կոորդինատների առանցքների վրա՝ արտածել 3:
19. Տրված է կետ, որն ընկած չէ կոորդինատների առանցքների վրա: Արտածել կոորդինատային քառորդի համարը, որտեղ ընկած է այդ կետը:
20. Արտածել `YES`, եթե տրված `(x,y)` կետը պատկանում է `(0,0),(0,-1),(-1,0)` կետերով կառուցված եռանկյանը, `NO`՝ հակառակ դեպքում:
21. Տրված են ուղղանկյան երեք գագաթների կոորդինատները: Ուղղանկյան կողմերը զուգահեռ են կոորդինատների առանցքներին: Արտածել չորրորդ կետի կոորդինատները:
22. Ներածել `n` թիվը: Արտածել 1, եթե դրա միավորների թվանշանը 3-ով մեծ է հարյուրավորների թվանշանից, հակառակ դեպքում՝ այդ թվի քառակուսին:
23. Տրված `n>7` բնական թվի համար գտնել այնպիսի `a` և `b` բնական թվեր, որ `3a+5b=n`:

4. Ցիկլեր

Հաճախ կարիք է լինում ծրագրի որոշակի հատված կատարել մեկից ավել անգամ՝ կազմակերպելով ցիկլ: C++ լեզվում կա ցիկլի կազմակերպման երեք կառույց՝ `while`, `for` և `do…while`: Դրանցից երկուսը՝ `while`-ը և `for`-ը, նախապայմանով ցիկլի կառույցներ են, քանի որ դրանք օգտագործելիս նախ ստուգվում է ցիկլն ավարտելու պայմանը, որից հետո միայն տեղի է ունենում ցիկլի հերթական կատարումը (իտերացիան): `do…while`-ը հետպայմանով ցիկլի կառույց է, քանի որ այն օգտագործելիս նախ տեղի է ունենում ցիկլի հերթական իտերացիան, որից հետո միայն ստուգվում է ցիկլն ավարտելու պայմանը:

`while` կառույցն ունի հետևյալ տեսքը՝

`while(`*պայման*`)`
    հրաման

պայման-ը տրամաբանական արտահայտություն է, *հրաման*-ը ցանկացած, այդ թվում՝ բաղադրյալ, հրաման է, որը կատարվում է ցիկլի յուրաքանչյուր իտերացիայի ժամանակ: `while` կառույցի աշխատանքի սկզբունքը հետևյալն է՝ նախ ստուգվում է *պայման*-ի արժեքը և եթե այն `true` է, ապա կատարվում է *հրաման*-ը և, այդ պրոցեսը նորից կրկնվում է, հակառակ դեպքում (երբ *պայման*-ի արժեքը `false` է)՝ ցիկլն ավարտվում է: Քանի որ C++ լեզվում գործում է անբացահայտ ձևափոխություն թվային տիպերից դեպի տրամաբանական տիպ (`0`-ն դառնում է `false`, իսկ մնացած թվերը՝ `true`), ապա *պայման*-ը կարող է լինել նաև թվաբանական արտահայտություն:

Ցիկլի `for` կառույցն ունի հետևյալ տեսքը՝

`for(`*արտ1*`;` *պայման*`;` *արտ2*`)`
    հրաման

արտ1-ը արտահայտություն է կամ փոփոխականների հայտարարություն, *պայման*-ը տրամաբանական արտահայտություն է, *արտ2*-ը արտահայտություն է, *հրաման*-ը ցանկացած (այդ թվում՝ բաղադրյալ), հրաման է, որը կատարվում է ցիկլի յուրաքանչյուր իտերացիայի ժամանակ: *հրաման*-ը կարող է նաև դատարկ լինել, այսինքն փակվող փակագծին կարող է հաջորդել կետ ստորակետ: `for` կառույցի աշխատանքի սկզբունքը հետևյալն է. մինչև ցիկլը սկսելը կատարվում է *արտ1*-ը: Ընդ որում, եթե *արտ1*-ը իրենից փոփոխականների հայտարարություն է, ապա այդ փոփոխականների տեսանելիության տիրույթը ցիկլի մարմինն է: *արտ1*-ը կատարելուց հետո սկսվում է կրկնվող պրոցեսը՝ ստուգվում է *պայման*-ի արժեքը և եթե այն `true` է, ապա կատարվում է *հրաման*-ը, այնուհետև՝ *արտ2*-ը, և այդ պրոցեսը նորից կրկնվում է, հակառակ դեպքում (երբ *պայման*-ի արժեքը `false` է)՝ ցիկլն ավարտվում է:

`do…while` կառույցն ունի հետևյալ տեսքը՝

```
do
   հրաման
while(պայման);
```

պայման-ը տրամաբանական արտահայտություն է, *հրաման*-ը ցանկացած (այդ թվում՝ բաղադրյալ), հրաման է, որը կատարվում է ցիկլի յուրաքանչյուր իտերացիայի ժամանակ: `do…while` կառույցի աշխատանքի սկզբունքը հետևյալն է՝ նախ կատարվում է *հրաման*-ը, որից հետո ստուգվում է *պայման*-ի արժեքը և եթե այն `true` է, ապա այդ պրոցեսը նորից կրկնվում է, հակառակ դեպքում (երբ *պայման*-ի արժեքը `false` է)՝ ցիկլն ավարտվում է: Այժմ ցիկլի օգնությամբ լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Արտածել տրված բնական թվի բոլոր բաժանարարները:

Լուծում՝ կներկայացնենք այս խնդրի լուծման երկու տարբերակ։

```
#include <iostream>

using namespace std;

int main()
{
    int n;
    cin>>n;
    for(int i=1;i<=n;++i)
        if(n%i==0)
            //i-ն հանդիսանում է n-ի բաժանարար
            cout<<i<<' ';
    cout<<endl;
    return 0;
}
```

Լուծման երկրորդ տարբերակ՝

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin>>n;
    cout<<1<<' '; //1-ը միշտ բաժանարար է
    if(n!=1)
        cout<<n<<' '; //թիվը ինքն իր բաժանարարն է
    int i=2;
    for(;i*i<n;++i)
        if(n%i==0) //i-ն և n/i-ն բաժանարար են
            cout<<i<<' '<<n/i<<' ';
```

```
    if(i*i==n) //այս դեպքը չենք դիտարկել
        cout<<i<<' ';
    cout<<endl;
    return 0;
}
```

Լինում են դեպքեր (օրինակ՝ ինչ-որ պայման տեղի ունենալու դեպքում), երբ կարիք է լինում անմիջապես ավարտել ցիկլի աշխատանքը կամ անմիջապես ավարտել ցիկլի ընթացիկ իտերացիան: `break` և `continue` հրամանները նախատեսված են այդ դեպքերի համար: Երբ ցիկլի ներսում կատարվում է `break` հրամանը, ապա ցիկլի կատարումն անմիջապես ավարտվում է: Երբ ցիկլի ներսում կատարվում է `continue` հրամանը, ապա անմիջապես ավարտվում է ցիկլի ընթացիկ իտերացիան և շարունակվում է ցիկլի կառույցի աշխատանքը՝ `for` կառույցի դեպքում կատարվում է *արտ2*-ը, իսկ մյուս երկու կառույցների դեպքում ստուգվում է ցիկլի ավարտի պայմանը: `break` և `continue` հրամանները ազդում են միայն հրամանն ընդգրկող ամենամոտ ցիկլի կառույցի վրա: Այժմ դիտարկենք հետևյալ խնդիրը, որի լուծման համար օգտագործելու ենք ներդրված ցիկլներ՝

Օրինակ 2. Արտածել տրված `[a,b]` միջակայքին պատկանող դրական ամբողջ սիմետրիկ թվերի միջին թվաբանականը:

Լուծում՝

```
#include <iostream>

using std::cin;
using std::cout;
using std::endl;
```

```
int main()
{
    int a,b;
    cin>>a>>b;
    int sum=0; //սիմետրիկ թվերի գումարն է
    int count=0; //սիմետրիկ թվերի քանակն է
    for(int i=a;i<=b;++i)
    {
        /*i-ն ընդունում է հատվածի ձախ ծայրակետից
        մինչև աջ ծայրակետն ընկած ամբողջ արժեքները*/
        int current=i;
        int currentR=0;
        /*currentR փոփոխականի մեջ ստանալու ենք
        հերթական թվի շրջված տարբերակը*/
        while(current)
        {
            currentR*=10;
            currentR+=current%10;
            current/=10;
        }
        if(i==currentR) //հերթական թիվը սիմետրի՞կ է
        {
            sum+=i;
            ++count;
        }
    }
    if(count) //եթե հատվածում սիմետրիկ թիվ կա
    {
        //հաշվում և արտածում ենք միջին թվաբանականը
        cout<<(double)sum/count<<endl;
    }
    else
    {
```

```
        /*արտածում ենք հաղորդագրություն այն մասին, որ
        սիմետրիկ թիվ չկա*/
        cout<<"There is no symmetric number"
            <<endl;
    }
    return 0;
}
```

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն՝ օգտագործելով ցիկլի կառույցներ:

1. Ի՞նչ կարտածի հետևյալ ծրագիրը.

ա)
```
#include <iostream>
using namespace std;
int main(){
    short b=0x543;
    int s=0;
    for(int i=0;i<16;++i,b/=2)
        s+=b%2?1:2;
    cout<<s<<endl;
    return 0;
}
```

բ)
```
#include <iostream>
using namespace std;
int main(){
    int c=7;
    while(c--){
        if(c%2==0){
            cout<<"a ";
            continue;
        }
        cout<<c<<' ';
    }
    return 0;
}
```

գ)
```
#include <iostream>
```

```
using namespace std;
int main(){
    char c='a'; bool b=true;
    for(;c<='z' && b;++c){
        switch(c){
            case 'b':
                ++c; break;
            case 'f':
                --c;
            case 'h':
                b=false; break;
            default:
                cout<<c<<' '; continue;
        }
        cout<<c<<' ';
    }
    return 0;
}
```

դ)
```
#include <iostream>
using namespace std;
int main(){
    int x,y;
    for (x=1,y=555;y>1;x*=2,y/=10){
        if(x==4) continue;
            cout<<"x="<<x<<" y="<<y<<endl;
    }
    return 0;
}
```

2. Տրված են `K` և `N(N>0)` ամբողջ թվերը: `N` անգամ արտածել `K` թիվը:
3. Ներածել `n` միանիշ թիվը: Արտածել `n`-ին չգերազանցող թվանշանները:
4. Ներածել `n` բնական թիվը: Արտածել `n`-ի՝ `1`-ից մինչև `9`-րդ աստիճանների արժեքները:

5. Հաշվել տրված բնական թվին չգերազանցող զույգ թվերի գումարը՝ առանց զույգությունը ստուգելու:
6. Տրված են A և B (A<B) ամբողջ թվերը:
   ա) Արտածել [A,B] հատվածին պատկանող 3-ին պատիկ առաջին թիվը:
   բ) Արտածել [A,B] հատվածին պատկանող բոլոր ամբողջ թվերի գումարը:
   գ) Աճման կարգով արտածել [A,B] հատվածին պատկանող ամբողջ թվերը և դրանց քանակը:
7. Տրված են երկու բնական x և y թվեր: Գտնել դրանց
   ա) ամենամեծ ընդհանուր բաժանարարը,
   բ) ամենափոքր ընդհանուր բազմապատիկը,
   գ) C_x^y, եթե $x \geq y$ կամ C_y^x, եթե $y \geq x$:
8. Տրված է N բնական թիվը: Արտածել.
   ա) 1+1/2+1/3+…+1/N գումարը (այն իրական թիվ է),
   բ) $N^2+(N+1)^2+(N+2)^2+\ldots+(2 \cdot N)^2$ գումարը,
   գ) 1.1·1.2·1.3·… արտադրյալը (արտադրիչների քանակը N է),
   դ) 1.1-1.2+1.3-… արտահայտության արժեքը (նշանափոխ գումարելիների քանակը N է, պայմանական if կառույցը չօգտագործել),
   ե) այդ թվի քառակուսին՝ օգտագործելով հետևյալ բանաձևը. $N^2=1+3+5+\ldots+(2N-1)$: Հերթական գումարելին ավելացնելիս արտածել գումարի ընթացիկ արժեքը (արդյունքում կարտածվեն 1-ից N բոլոր թվերի քառակուսիները):
9. Տրված են A իրական և N բնական թվերը: Արտածել.
   ա) A-ի N աստիճանը,
   բ) A-ի բոլոր աստիճանները՝ 1-ից մինչև N (մեկ ցիկլի մի-

ջոցով),

գ) $1+A+A^2+A^3+\ldots+A^N$ գումարը (մեկ ցիկլի միջոցով),

դ) $1-A+A^2-A^3+\ldots+(-1)^N A^N$ արտահայտության արժեքը (մեկ ցիկլի միջոցով և առանց պայմանական `if` կառույցը օգտագործելու),

ե) $A(A+1)\ldots(A+N-1)$ արտադրյալը:

10. Տրված է `N` բնական թիվը: Արտածել.

ա) $N!=1\cdot 2\cdot\ldots\cdot N$ արտահայտության արժեքը (`N`-ի ֆակտորիալը),

բ) $1!+2!+3!+\ldots+N!$ գումարը (մեկ ցիկլի միջոցով),

գ) $(1+1/1^2)(1+1/2^2)\ldots(1+1/n^2)$ արտադրյալը,

դ) $1/\sin 1+1/(\sin 1+\sin 2)+\ldots+1/(\sin 1+\ldots \sin N)$ գումարը:

11. Ներածելով `a,d,n` արժեքները՝ արտածել $a, a+d, \ldots, a+(n-1)d$ թվաբանական պրոգրեսիայի անդամների գումարը:

12. Տրված են `A` իրական և `N` բնական թվերը: Արտածել հետևյալ գումարը.

ա) $1/a+1/a(a+1)+\ldots+1/a(a+1)\ldots(a+N)$

բ) $1/a+1/a^2+1/a^4+\ldots+1/a^{2^N}$

13. Տրված են `N` բնական թիվը և $A, B (A<B)$ իրական թվերը թվային առանցքի վրա: `[A,B]` հատվածը տրոհված է `N` հավասար մասերի: Արտածել.

ա) յուրաքանչյուր հատվածի `H` երկարությունը և $A, A+H, A+2\cdot H, A+3\cdot H, \ldots, B$ արժեքները,

բ) յուրաքանչյուր հատվածի `H` երկարությունը և $A, A+H, A+2\cdot H, A+3\cdot H, \ldots, B$ կետերում $F(X)=1-\sin(X)$ ֆունկցիայի արժեքները:

14. Տրված է `N` բնական թիվը: A_K իրական թվերի հաջորդականությունը սահմանվում է հետևյալ կերպ.

ա) $A_1=2,\ A_K=2+1/A_{K-1},\ K=2,3,\ldots$

բ) $A_1=1,\ A_K=(A_{K-1}+1)/K,\ K=2,3,\ldots$

գ) $A_1=1,\ A_2=1,\ A_K=A_{K-1}+A_{k-2},\ K=3,4,\ldots$ (ֆիբոնաչիի հաջորդականություն)

Արտածել $A_1, A_2, \ldots, A_N$ տարրերը:

15. Արտածել տրված բնական թվի
    ա) թվանշանների գումարը, քանակը, արտադրյալը,
    բ) կենտ թվանշաններից ամենափոքրը,
    գ) 4-ից մեծ կենտ թվանշանների գումարը,
    դ) 7-ից փոքր զույգ թվանշանների արտադրյալը,
    ե) ամենամեծ և ամենափոքր թվանշանների տարբերության քառակուսին:
16. Տեղերով փոխել տրված թվի առաջին և վերջին թվանշանները: Օրինակ՝ 8547-ից պետք է ստացվի 7548:
17. Տրված է բնական թիվ: Արտածել YES, եթե
    ա) թվի թվանշանների մեջ կա 3 թվանշան,
    բ) թվի թվանշանների մեջ չկա 5 թվանշան,
    գ) թվի թվանշանները աճման կարգով են դասավորված,
    դ) թվի թվանշանները նվազման կարգով չեն դասավորված,
    ե) թվի թվանշանների գումարը մեծ է քսանից,
    զ) թվի թվանշանների արտադրյալը փոքր է երեսունից:

    Հակառակ դեպքում արտածել NO:
18. Արտածել տրված բնական թվի
    բ) բոլոր բաժանարարների գումարը,
    գ) բոլոր բաժանարարների գումարի և դրանց քանակի քանորդը,
    դ) պարզ արտադրիչների քանակը,
    ե) ներկայացումը պարզ արտադրիչների արտադրյալի տեսքով:
19. Պարզել, արդյո՞ք տրված բնական թիվը 3-ի որևէ ամբողջ աստիճան է, թե ոչ:
20. Տրված $n>0$ ամբողջ թվի համար արտածել

ա) 2-ի ամենամեծ աստիճանը, որը չի գերազանցում `n`-ը,

բ) 2-ի ամենափոքր աստիճանը, որը գերազանցում է `n`-ը:

21. Տրված բնական թվի համար արտածել `YES`, եթե

ա) այն որևէ թվի ֆակտորիալ է,

բ) այն ֆիբոնաչիի թիվ է,

գ) այն կատարյալ թիվ է,

դ) այն պարզ թիվ է,

ե) այն սիմետրիկ թիվ է:

Հակառակ դեպքում արտածել `NO`:

22. Տրված `n` բնական թվի համար արտածել ֆիբոնաչիի հաջորդականության `n`-րդ անդամը:

23. Տրված `n` բնական թվի համար արտածել ֆիբոնաչիի հաջորդականության 1-ից մինչև `n`-րդ անդամների գումարը:

24. Ստուգել՝ տրված թվի թվանշանները ձախից աջ դիտարկելիս արդյո՞ք կազմում են նվազող հաջորդականություն, թե ոչ: Օրինակ՝ `76431` թվի համար պատասխանը դրական է, իսկ `6331` և `9782` թվերի համար՝ բացասական:

25. Արտածել տրված բնական թիվը՝ շրջելով այն և նրա ամեն մի թվանշանից հետո ավելացնելով 0: Օրինակ՝ `125`-ի դեպքում պետք է արտածել `502010`:

26. Արտածել `YES`, եթե տրված բնական թիվը հավասար է իր թվանշանների գումարի կրկնապատիկին, `NO`՝ հակառակ դեպքում:

27. Արտածել բոլոր այն եռանիշ թվերը, որոնց առաջին թվանշանը ջնջելուց և ստացված թիվը `7`-ով բազմապատկելուց հետո ստացվում է սկզբնական թիվը:

28. Արտածել բոլոր այն քառանիշ թվերը, որոնցում կրկնվող թվանշան չկա, և առաջին ու վերջին երկու թվանշաններից կազմված թվերի տարբերությունը հավասար է այդ թվի թվանշանների գումարին:

29. Գրել ծրագիր, որը տրված n(2<n<100) բնական թվի համար արտածում է բնական թվերի հաջորդականություն՝ հետևյալ կանոններով.
    - հաջորդականության առաջին թիվը n թիվն է,
    - հաջորդականության յուրաքանչյուր հերթական թիվ ստացվում է նախորդ p թվից և հավասար է 3p+1, եթե p-ն կենտ է և հավասար է p/2, եթե p-ն զույգ է:
    - ծրագիրը պետք է ավարտի իր աշխատանքը, եթե հաջորդականության հերթական տարրը հավասար է 1-ի:

    Օրինակ՝ n=7 դեպքում պետք է արտածել հետևյալ հաջորդականությունը՝ 7 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
30. Արտածել տրված բնական թվի 16-ական ներկայացումը:
31. Արտածել տրված բնական թվի 2-ական ներկայացման մեջ 1-երի քանակը:
32. Տրված են x և p ոչ բացասական թվերը (17>p>1): Արտածել x թվի p-ական համակարգում ներկայացման թվանշանները հակառակ կարգով:
33. Տրված է ամբողջ կողմերով (n և m) ուղղանկյուն: Անհրաժեշտ է այդ ուղղանկյունը ամբողջությամբ տրոհել նվազագույն թվով քառակուսիների: Արտածել այդ տրոհման քառակուսիների քանակը:
34. Տրված n բնական թվի համար արտածել 10^i,i=0,1,2,… թվերը փոքրից մեծ իրար կցելիս ստացված թվանշանների հաջորդականության n-րդ թվանշանը:
35. Տրված n բնական թվի համար արտածել բոլոր բնական թվերը փոքրից մեծ իրար կցելիս ստացված թվանշանների հաջորդականության n-րդ թվանշանը:

Ներածում ցիկլի մեջ

36. Ներածել թվեր: Արտածել դրանցից բացասականների քառակուսիների արտադրյալը, քանի դեռ այն մեծ չէ 400-ից:
37. Ներածել ամբողջ թվերի հաջորդականություն՝ մինչև 0-ի հանդիպելը: Պարզել.
    ա) հաջորդականությունը աճող է, թե ոչ,
    բ) հաջորդականությում արդյո՞ք կա իրար հավասար երկու հարևան թվերի գոնե մեկ զույգ, թե ոչ,
    գ) հաջորդականությունը նշանափոխ է, թե ոչ:
38. Ներածել տրված քանակով ամբողջ թվերի հաջորդականություն: Արտածել
    ա) բացարձակ արժեքով ամենամեծ թվի թվանշանների քանակը,
    բ) 7-ի վրա առանց մնացորդի բաժանվող թվերի միջին թվաբանականը:
39. Ներածել տրված քանակով իրական թվերի հաջորդականություն: Արտածել
    ա) 0-ին ամենամոտ թիվը,
    բ) այդ թվերի քառակուսիների գումարը:
40. Ներածել $a_1, a_2, \ldots, a_n$ իրական թվերի հաջորդականությունը և արտածել հետևյալ արտահայտության արժեքը՝ $n \cdot a_1 - (n-1) \cdot a_2 + (n-2) \cdot a_3 - \ldots + (-1)^{n-1} \cdot 1 \cdot a_n$:

Վերջավոր և անվերջ գումարներ ու արտադրյալներ

41. Տրված է N բնական թիվը: Օգտագործելով մեկ ցիկլ՝ հաշվել հետևյալ գումարը. $1+1/1!+1/2!+1/3!+\ldots+1/N!$ (ստացված թիվը e հաստատունի մոտավոր արժեքն է):
42. Ներածել X իրական և N բնական թվերը: Արտածել հետևյալ արտահայտության արժեքը՝
    ա) $1+X+X^2/2!+\ldots+X^N/N!$ (e^X-ի մոտավոր արժեքը),

բ) $X-X^3/3!+X^5/5!-\ldots+(-1)^N\cdot X^{2N+1}/(2N+1)!$ (sinX-ի մոտավոր արժեքը),

գ) $1-X^2/2!+X^4/4!-\ldots+(-1)^N\cdot X^{2N}/(2N)!$ (cosX-ի մոտավոր արժեքը):

43. Տրված են X իրական ($|X|<1$) և N բնական թվերը: Արտածել հետևյալ արտահայտության արժեքը՝

ա) $X-X^2/2+X^3/3-\ldots+(-1)^{N-1}\cdot X^N/N$ (lnX-ի մոտավոր արժեքը),

բ) $X-X^3/3+X^5/5-\ldots+(-1)^N\cdot X^{2N+1}/(2N+1)$ (arctgX-ի մոտավոր արժեքը),

գ) $X+1!!\cdot X^3/(2!!\cdot 3)+3!!\cdot X^5/(4!!\cdot 5)+\ldots+(2N-1)!!\cdot X^{2N+1}/((2N)!!\cdot(2N+1))$ (arcsinX-ի մոտավոր արժեքը),

դ) $1+X/2-1!!\cdot X^2/(4!!)+3!!\cdot X^3/6!!-\ldots+(-1)^{N-1}\cdot(2N-3)!!\cdot X^N/(2N)!!$ (1+X-ի մոտավոր արժեքը),

որտեղ k!!-ը [1,k] միջակայքի բոլոր զույգ թվերի արտադրյալն է (զույգ k-երի համար) կամ [1,k] միջակայքի բոլոր կենտ թվերի արտադրյալն է (կենտ k-երի համար):

44. Հաշվել հետևյալ վերջավոր գումարները.

ա) $\sum_{i=1}^{n} i$ դ) $\sum_{i=0}^{n} \frac{\sin x^i}{3^i(i+1)!}$ զ) $\sum_{i=1}^{n} (-1)^{i+1}\frac{2^{2i}+5}{(2i)!}$

բ) $\sum_{i=0}^{n} \frac{a^i}{i!}$ ե) $\sum_{i=1}^{n} (-1)^i\frac{a^{2i}}{2^i(2i)!}$ է) $\sum_{i=1}^{n} (-1)^i\frac{x^{i+1}}{(3i)!+2^{i+1}}$

գ) $\sum_{i=0}^{n} \frac{a^{2i}-3}{(2i+1)!}$ ը) $\sum_{i=1}^{n}\left(1+\frac{\cos^i x}{(3i-1)!}\right)$

45. Հաշվել հետևյալ վերջավոր արտադրյալները.

ա) $\prod_{i=0}^{n}(3i+1)$ դ) $\prod_{i=0}^{n}(-1)^{i+1}\frac{a^{2i+1}}{(2i+1)!}$

բ) $\prod_{i=1}^{n}(-1)^i\frac{2^{i+1}}{(i+1)!}$ ե) $\prod_{i=1}^{n}\frac{1+3^{2i}}{(3i+2)!}$

գ) $\prod_{i=0}^{n}(-1)^i\frac{a^{3i}2^i}{(2i)!}$

46. Տրված ճշտությամբ հաշվել հետևյալ անվերջ գումարները.

ա) $\sum_{i=0}^{\infty}\frac{(-2)^i}{(2i)!}$ ե) $\sum_{i=1}^{\infty}(-1)^{i-1}\frac{x^{2i+1}}{3^i(3i+2)!}$

բ) $\sum_{i=0}^{\infty}\frac{3^{2i}}{(2i+1)!}$ զ) $\sum_{i=0}^{\infty}\frac{1}{4^i+5^{i+2}}$

գ) $\sum_{i=0}^{\infty}(-1)^i\frac{x^{2i}}{2^i(3i-1)!}$ է) $\sum_{i=1}^{\infty}\frac{\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{i+1}}{i!}$

դ) $\sum_{i=1}^{\infty}(-1)^{i+1}\frac{x^i}{i(i+1)(i+2)}$ ը) $\sum_{i=0}^{\infty}\left(1+\frac{\sin(ix)}{i!}\right)$

Ներդրված ցիկլեր

47. Ի՞նչ կարտածի հետևյալ ծրագիրը.

```
#include <iostream>
using namespace std;
int main(){
    for(int i=0;i<4;++i)
        for(int j=0;j<i;j+=2){
            cout<<j<<' '<<i<<endl;
        }
    return 0;
}
```

48. Արտածել 1-ից մեծ այն ամենափոքր թիվը, որը հավասար է իր քառակուսու թվանշանների գումարին:

49. Արտածել 100-ից փոքր բոլոր այն թվերը, որոնց քառակուսիները սիմետրիկ են (այսինքն աջից ու ձախից կարդալիս ստացվում է նույն թիվը):
50. n թվանշան ունեցող բնական թիվը կոչվում է Արմսթրոնգի թիվ, եթե նրա թվանշանների n աստիճանների գումարը հավասար է այդ թվին (օրինակ՝ $153=1^3+5^3+3^3$): Արտածել բոլոր երկնիշ, եռանիշ և քառանիշ Արմսթրոնգի թվերը:
51. Ներածել բնական թվեր՝ մինչև 0-ի հանդիպելը և արտածել դրանցում կատարյալ թվերի քանակը:
52. Ներածել տրված քանակով բնական թվեր և արտածել դրանցում
    ա) 7-ով սկսվող թվերի քանակը,
    բ) 5 թվանշանը պարունակող թվերի քանակը,
    գ) պարզ թվերի քանակը,
    դ) հանդիպող նվազագույն կատարյալ թիվը:
53. Արտածել տրված n թիվը չգերազանցող բոլոր այն բնական թվերը, որոնք առանց մնացորդի բաժանվում են իրենց թվանշանների գումարի վրա:
54. Տրված n բնական թվի համար արտածել
    ա) մինչև այդ թիվը եղած և n-ի հետ փոխադարձաբար պարզ թվերի քանակը,
    բ) [1,n] միջակայքի բոլոր պարզ թվերի միջին թվաբանականը,
    գ) այդ թվին ամենամոտ պարզ թիվը,
    դ) YES, եթե այն ունի a^b տեսքը և NO՝ հակառակ դեպքում:
55. Ներածել տրված քանակով բնական թվեր և արտածել բոլոր այն թվերը, որոնց
    ա) թվանշանների քանակը կենտ է,
    բ) թվանշանների գումարը բաժանվում է 7-ի,
    գ) 7-ական ներկայացման թվանշանների գումարը մեծ է 10-ից,

դ) երկուական ներկայացման մեջ 1-երի քանակը զույգ է,
ե) թվանշանների գումարը բաժանվում է թվանշանների քանակի վրա:

56. Ներածել տրված քանակով բնական թվեր և արտածել
ա) այդ թվերի երկուական ներկայացումներում `1`-երի և `0`-ների քանակների տարբերությունները,
բ) այդ թվերի `3`-ական ներկայացումներում `2`-ների քանակները,
գ) բոլոր այն թվերի գումարը, որոնք որևէ բնական թվի քառակուսին են:

57. Երկու պարզ թվեր կոչվում են զույգ կազմող, եթե դրանց տարբերությունը 2 է (օրինակ՝ `5` և `3`, `101` և `103`): Արտածել `1000`-ից փոքր բոլոր զույգ կազմող պարզ թվերը:

58. Տրված է `n` բնական թիվը: Հաշվել նրա թվանշանների գումարը, ապա ստացված թվի թվանշանների գումարը և այդպես շարունակ, մինչև որ ստացվի միանիշ թիվ: Ընթացքում արտածել ստացված գումարները:

59. Տրված `n` բնական թվի համար արտածել `[1,n]` միջակայքի բոլոր պարզ թվերը: Առաջարկվում է այս խնդիրը լուծելու համար ծանոթանալ և օգտագործել «Էրատոսթենեսի մաղ» (`Sieve of Eratosthenes`) անունով ալգորիթմը:

5. Միաչափ զանգվածներ

Զանգվածը նույնատիպ տարրերի հաջորդականություն է, որի տարրերը հաջորդաբար են դասավորվում հիշողության մեջ: C++ լեզվում հնարավոր է ստեղծել ստատիկ և դինամիկ զանգվածներ: Ստատիկ զանգվածի տարրերի քանակը պետք է լինի կոմպիլյացիայի ընթացքում հաշվարկվող արժեք: Ստատիկ զանգվածի նկարագրման ընդհանուր տեսքը հետևյալն է՝

տիպ անուն[*չափ*] *սկզբնարժեքավորում*

տիպ-ը զանգվածի տարրերի տիպն է, *անուն*-ը զանգվածի անունն է, որը ցանկացած նույնարկիչ է, *չափ*-ը կոմպիլյացիայի ընթացքում հաշվարկվող ամբողջ տիպի արտահայտություն է, որի արժեքը զանգվածի տարրերի քանակն է, իսկ *սկզբնարժեքավորում*-ը օգտագործվում է զանգվածի տարրերը սկզբնարժեքավորելու համար և կարող է բացակայել: *սկզբնարժեքավորում*-ը ունի հետևյալ տեսքը՝

={ *տարրերի արժեքների ցուցակ*}

Չնավոր փակագծերում նշվում է զանգվածի տարրերի սկզբնական արժեքները, որոնք բաժանվում են ստորակետով: Ընդ որում, եթե տրված արժեքների քանակը ավելի քիչ է զանգվածի տարրերի քանակից, ապա մնացած տարրերը սկզբնարժեքավորվում են համապատասխան տիպի լռելյայն արժեքով (որը C++-ի ներդրված տիպերի դեպքում հավասար է 0-ի): Ստորև բերված են ստատիկ զանգվածի նկարագրման մի քանի օրինակ՝

- ```
  /*10 տարր պարունակող int տիպի զանգված, որի
  տարրերը սկզբնարժեքավորված չեն*/
  int A[10];
  ```
  ```

- /*3 տարրանոց double տիպի զանգված, որի տարրերը սկզբնարժեքավորում ենք համապատասխանաբար 0.5, 0.6 և 0.7 արժեքներով*/
  ```
  const int n=2;
  double A[n+1]={0.5,0.6,0.7};
  ```
- /*1000 տարր պարունակող int տիպի զանգված, որի տարրերը սկզբնարժեքավորում ենք 0-ներով*/
  ```
  int A[1000]={0};
  ```
- /*5 տարր պարունակող int տիպի զանգված, որի տարրերը սկզբնարժեքավորում ենք 1-ից 5 թվերով*/
  ```
  int A[]={1,2,3,4,5};
  ```

Զանգվածի տարրերին կարող ենք դիմել հետևյալ եղանակով՝

անուն[*ինդեքս*]

անուն-ը զանգվածի անունն է, *ինդեքս*-ը ամբողջ տիպի արտահայտությունն է, որի արժեքը զանգվածի այն տարրի հերթական համարն է (համարակալումը սկսվում է 0-ից), որին ցանկանում ենք դիմել: Ստատիկ զանգվածի անունը ցուցիչ հաստատուն է (այսինքն՝ *տիպ** const տիպի ցուցիչ), որը ցույց է տալիս զանգվածի առաջին տարրի վրա: Այժմ, օգտագործելով ստատիկ զանգված, լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Տրված է իրական թվերի հաջորդականություն: Արտածել այդ հաջորդականության ամենաշատ կրկնվող տարրը:

Լուծում՝

```
#include <iostream>

using namespace std;

int main()
{
```

```
const int n=10; //հաջորդականության չափը
double a[n];
for(int i=0;i<n;++i)
{
    //ներածում ենք թվերի հաջորդականությունը
    cin>>a[i];
}
//ամենաշատ կրկնվող տարրի կրկնումների քանակը
int max=0;
double element; //ամենաշատ կրկնվող տարրը
/*պարունակում է ինֆորմացիա դիտարկված տարրերի
մասին*/
bool processed[n]={0};
for(int i=0;i<n;++i)
{
    if(processed[i])
    {
        //բաց ենք թողնում դիտարկված տարրերը
        continue;
    }
    /*հաշվում ենք, թե i-րդ տարրը քանի անգամ է
    հանդիպում իրենից հետո*/
    int count=0;
    for(int j=i+1;j<n;++j)
    {
        if(a[i]==a[j])
        {
            /*j-րդ տարրը համարում ենք
            դիտարկված*/
            processed[j]=true;
            ++count;
        }
    }
```

```
            if(max<count)
            {
                //գտել ենք ավելի հաճախ կրկնվող տարր
                element=a[i]; max=count;
            }
        }
        cout<<element<<endl;
        return 0;
}
```

Դինամիկ զանգված օգտագործելու անհրաժեշտությունն առաջանում է այն դեպքում, երբ զանգվածի տարրերի քանակը հայտնի է դառնում ծրագրի աշխատանքի ընթացքում: Դինամիկ զանգված կարելի է ստեղծել օգտագործելով `new` գործողությունը, որը հիշողությունից հատկացնում է զանգվածի տարրերը պահելու համար անհրաժեշտ տիրույթ և վերադարձնում է ցուցիչ զանգվածի առաջին (`0` ինդեքսով) տարրի վրա՝

new տիպ`[`*չափ*`]`

տիպ-ը դինամիկ զանգվածի տարրերի տիպն է, *չափ*-ը ամբողջ տիպի արտահայտություն է, որի արժեքը հավասար է ստեղծվող դինամիկ զանգվածի տարրերի քանակին: Ի տարբերություն ստատիկ զանգվածի, որի համար հատկացված հիշողությունն ազատվում է ավտոմատ կերպով՝ դինամիկ զանգվածի դեպքում պետք է զանգվածի հետ աշխատանքն ավարտելուց հետո ազատել հատկացված հիշողությունը: Հիշողությունն ազատելու համար պետք է օգտագործել `delete` գործողությունը՝

`delete []` *ցուցիչ*

ցուցիչ-ը զանգվածի առաջին տարրի հասցեն է: Օրինակ՝ կարող ենք ստեղծել և օգտագործել տրված չափի (չափը մուտքագրվում է ծրագրի աշխատանքի ընթացքում) `int` տիպի դինամիկ զանգված հետևյալ եղանակով՝

```
int n;
cin>>n;
int* A=new int[n];//ստեղծում ենք դինամիկ զանգված
/*օգտագործում ենք A զանգվածը*/
delete [] A;//ազատում ենք հատկացված հիշողությունը
```

Այժմ, օգտագործելով դինամիկ զանգված, լուծենք հետևյալ խնդիրը՝

Օրինակ 2. Տրված են իրական թվերի $a_0,\dots,a_{n-1}$ և $b_0,\dots,b_{m-1}$ չնվազող հաջորդականությունները: Ստանալ $c_0,c_1,\dots,c_{n+m-1}$ չնվազող հաջորդականությունը՝ կազմված տրված հաջորդականությունների բոլոր տարրերից:

Լուծում՝

```
#include <iostream>

using namespace std;

int main()
{
    int n,m; //հաջորդականությունների չափերը
    cin>>n>>m;
    //ստեղծում և ներածում ենք հաջորդականությունները
    double* a=new double[n];
    double* b=new double[m];
    double* c=new double[n+m];
    for(int i=0;i<n;++i)
        cin>>a[i];
    for(int i=0;i<m;++i)
        cin>>b[i];
    int i,j,k;
    i=j=k=0; //հաջորդականությունների ինդեքսները
    do{
        if(a[i]<b[j])
```

```
            c[k++]=a[i++];
        else
            if(a[i]>b[j])
                c[k++]=b[j++];
            else
            {
                c[k++]=a[i++];
                c[k++]=b[j++];
            }
    }while (i<n && j<m);
    /*ցիկլից հետո հաջորդականություններից մեկն ամբողջությամբ կգրվի նոր զանգվածում, իսկ մյուսի մնացած տարրերը պետք է շարունակել գրել*/
    while (i<n)
        c[k++]=a[i++];
    while (j<m)
        c[k++]=b[j++];
    //արտածում ենք արդյունքը
    for(int i=0;i<n+m;++i)
        cout<<c[i]<<' ';
    //ազատում ենք դինամիկ հիշողությունը
    delete []a;
    delete []b;
    delete []c;
    return 0;
}
```

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն՝ օգտագործելով ստատիկ կամ դինամիկ զանգված:

1. Տրված է ամբողջ թվերի հաջորդականություն: Արտածել այդ հաջորդականության
   ա) `[-5,10]` միջակայքին պատկանող տարրերի քանակը,
   բ) դրական և կենտ տարրերի քանակը,
   գ) `5`-ից մեծ զույգ տարրերի գումարը,

դ) 15-ից փոքր և 3-ին պատիկ տարրերի արտադրյալը,
ե) բոլոր տարրերի միջին թվաբանականը,
զ) բացասական և 7-ին պատիկ տարրերի միջին թվաբանականը,
է) 3-ով սկսվող թվերի քանակը,
ը) բոլոր զույգ դրական տարրերը, այնուհետև բոլոր բացասական տարրերը:

2. Տրված է իրական թվերի հաջորդականություն: Արտածել այդ հաջորդականության
ա) այն տարրերի քանակը, որոնք փոքր են իրենց կարգահամարի քառակուսուց,
բ) բոլոր տարրերի միջին թվաբանականի և տարրերի տարբերությունները,
գ) այն տարրերի քանակը, որոնք փոքր են իրենց ձախ և աջ հարևանների կիսագումարից,
դ) մեծագույն տարրը և նրա կարգահամարը,
ե) մեծագույն և փոքրագույն տարրերի տարբերությունը,
զ) առաջին մեծագույն և վերջին փոքրագույն տարրերի միջև գտնվող անդամների գումարը,
է) մեծագույն և մեծությամբ երկրորդ տարրերը,
ը) դրական տարրերից փոքրագույնի արժեքը (եթե դրական տարր չկա, արտածել համապատասխան հաղորդագրություն):

3. Արտածել տրված ամբողջ թվերի հաջորդականությունը ցիկլիկ տեղաշարժված դեպի աջ 4 տարրով:

4. Տրված է իրական թվերի հաջորդականություն: Արտածել YES, եթե
ա) դրանում դրական տարրերի քանակը ավելի մեծ է բացասական տարրերի քանակից,
բ) այն սիմետրիկ հաջորդականություն է,

գ) այն թվաբանական պրոգրեսիա է,

դ) այն երկրաչափական պրոգրեսիա է:

Հակառակ դեպքում արտածել `NO`:

5. Գտնել տրված իրական թվերի հաջորդականության փոքրագույն անդամը և տեղերով փոխել վերջին տարրի հետ:
6. Տրված է ամբողջ թվերի հաջորդականություն: Արտածել այդ հաջորդականության

ա) առաջին միանիշ տարրը,

բ) `5`-ին պատիկ վերջին տարրը,

գ) վերջին երկնիշ տարրը և դրան հաջորդող կենտ տարրերի քանակը,

դ) առաջին զույգ տարրը և դրան նախորդող տարրերի գումարը,

ե) `7`-ից մեծ վերջին տարրը և դրան նախորդող զույգ տարրերի արտադրյալը,

զ) մինչև առաջին բացասական տարրը եղած այն տարրերի քանակը, որոնք պատկանում են `[0,20]` միջակայքին,

է) `[5,24]` միջակայքում գտնվող առաջին տարրը և դրան հաջորդող տարրերի միջին թվաբանականը,

ը) առաջին `0`-ին հաջորդող `1`-ով սկսվող թվերի գումարը (եթե հաջորդականությունը `0` չի պարունակում՝ արտածել `-1`):

7. Տեղերով փոխել տրված իրական թվերի հաջորդականության առաջին և վերջին բացասական տարրերը (համարել, որ հաջորդականությունում կա առնվազն մեկ բացասական թիվ):
8. Տրված է իրական թվերի հաջորդականություն: Արտածել այդ հաջորդականության

ա) կրկնվող տարրերի քանակը,

բ) այն տարրերի քանակը, որոնք հաջորդականության մեջ հանդիպում են ճիշտ `k` անգամ, որտեղ `k`-ն տրված

բնական թիվ է,

գ) այն տարրերի քանակը, որոնք բավարարում են $2^k<a_k<k!$ պայմանին, որտեղ a_k-ն k-րդ տարրն է, իսկ k-ն նրա կարգահամարն է,

դ) հաջորդող զրոների ամենաերկար ենթահաջորդականության երկարությունը,

ե) աճման կարգով դասավորված իրար հաջորդող տարրերի ամենաերկար ենթահաջորդականության երկարությունը:

9. Տրված ամբողջ թվերի հաջորդականության համար արտածել, թե քանի անգամ է կրկնվում մեծագույն տարրը:

10. Տրված է բնական թվերի հաջորդականություն: Արտածել այդ հաջորդականության

ա) այն տարրերի քանակը, որոնց թվանշանների գումարը հավասար է տրված c թվին,

բ) 2-ի աստիճան հանդիսացող տարրերի արտադրյալը,

գ) որևէ թվի ֆակտորիալ հանդիսացող տարրերի միջին թվաբանականը,

դ) կատարյալ թվերի գումարը,

ե) պարզ թվերի արտադրյալը,

զ) Ֆիբոնաչիի թվերի քանակը,

է) սիմետրիկ թվերի միջին թվաբանականը,

ը) իրար հաջորդող սիմետրիկ թվերի ամենաերկար ենթահաջորդականության երկարությունը,

թ) իրար հաջորդող տարրերի այն ամենաերկար ենթահաջորդականությունը, որի անդամների գումարը 19-ին պատիկ է,

ժ) իրարից տարբեր զույգ թվերի քանակը,

ի) այն տարրերը, որոնք որևէ բնական թվի 1-ից մեծ բնական աստիճան են:

11. Տրված է բնական թվերի հաջորդականություն: Արտածել այդ հաջորդականության
    ա) տարրերի ֆակտորիալների գումարը,
    բ) տարրերի n աստիճանների միջին թվաբանականը, որտեղ n-ը տրված բնական թիվ է,
    գ) հարևան տարրերի ամենամեծ ընդհանուր բաժանարարների արտադրյալը,
    դ) հարևան տարրերի ամենափոքր ընդհանուր բազմապատիկների գումարը,
    ե) վերջին միանիշ տարրը և դրան նախորդող պարզ տարրերի արտադրյալը,
    զ) առաջին Ֆիբոնաչիի թիվը և դրան հաջորդող երկնիշ տարրերի միջին թվաբանականը,
    է) առաջին սիմետրիկ թիվը և դրան նախորդող բացասական տարրերի քանակը,
    ը) վերջին կատարյալ տարրը և դրան հաջորդող զույգ տարրերի արտադրյալը:
12. Պարզել, թե տրված բնական թվերի հաջորդականության տարրերի գրառման մեջ որ թվանշանն է մասնակցում առավելագույն քանակով:
13. Դասավորել տրված իրական թվերի հաջորդականության անդամները նվազման կարգով:
14. Դասավորել տրված ամբողջ թվերի հաջորդականության անդամներն անյպես, որ
    ա) սկզբում լինեն բացասականները,
    բ) վերջում լինեն կենտերը:
15. Տրված դրական ամբողջ թվերի հաջորդականության զրոյից տարբեր տարրերը բերել զանգվածի սկիզբ՝ ըստ իրենց հանդիպման հաջորդականության, իսկ զրոները տանել զանգվածի վերջ՝ չօգտագործելով օժանդակ զանգված:

16. Տրված իրական թվերի հաջորդականությունից հեռացնել բոլոր զրոներն ու խտացնել այն՝ չօգտագործելով օժանդակ զանգված:
17. Տրված իրական թվերի հաջորդականությունից հեռացնել k-րդ պարզ տարրն ու խտացնել այն՝ չօգտագործելով օժանդակ զանգված: Եթե զանգվածում պարզ տարր չկա, տալ համապատասխան հաղորդագրություն:
18. Արտածել տրված իրական թվերի հաջորդականության ամենաերկար աճող ենթահաջորդականության երկարությունը:
19. Տրված է բնական թվերի $a_0,\dots,a_{n-1}$ հաջորդականությունը: Կառուցել $b_0,\dots,b_{n-1}$ հաջորդականությունը, որտեղ
    ա) b_i-ն a_i-ի բաժանարարների գումարն է,
    բ) b_i-ն $a_0,\dots,a_i$-ի միջին թվաբանականն է,
    գ) b_i-ն $a_0,\dots,a_i$-ի ամենափոքր ընդհանուր բազմապատիկն է,
    դ) b_i-ն $a_0,\dots,a_i$-ի ամենամեծ ընդհանուր բաժանարարն է,
    ե) b_i-ն a_i-ի փոքրագույն թվանշանն է,
    զ) b_i-ն a_i-ի պարզ բաժանարարների քանակն է,
    է) $b_i=1$, եթե $a_0,\dots,a_i$-ում կան իրար հավասար տարրեր, և $b_i=0$՝ հակառակ դեպքում,
    ը) b_i-ն a_i-ում զրոյական բիթերի քանակն է,
    թ) b_i-ն a_i-ն չգերազանցող պարզ թվերի քանակն է,
    ժ) b_i-ն a_i-ն չգերազանցող սիմետրիկ թվերի քանակն է,
    ի) $b_i=a_i!$:
20. Կառուցել տրված բնական թվերի $a_0,\dots,a_{n-1}$ հաջորդականության $b_0,\dots,b_{k-1}$ ենթահաջորդականությունը ($k\le n$), որը բաղկացած է հետևյալ պայմանին բավարարող տարրերից.
    ա) a_i-ն մեծ է տրված թվից,
    բ) a_i-ն 4-ին պատիկ թիվ է,
    գ) a_i-ի երկուական ներկայացման մեջ մեկերի քանակը պարզ թիվ է,

դ) a_i-ի բաժանարարների քանակը մեծ է կամ հավասար տրված թվից,

ե) a_i-ի առաջին և վերջին բայթերում մեկերի քանակները հավասար են,

զ) a_i-ն սիմետրիկ թիվ է,

է) a_i-ն կատարյալ թիվ է,

ը) $a_i=i!$,

թ) a_i-ն մեծ է $a_0,\ldots,a_{n-1}$ հաջորդականության մեծագույն տարրի կեսից:

21. Տրված բնական թվերի հաջորդականությունից կառուցել դրա եռանիշ տարրերի կարգահամարներից կազմված նոր հաջորդականություն:
22. Տրված ամբողջ թվերի հաջորդականությունից կառուցել դրա բացասական տարրերի կարգահամարներից կազմված նոր հաջորդականություն:
23. Տրված են իրական թվերի $a_0,\ldots,a_{n-1}$ և $b_0,\ldots,b_{m-1}$ հաջորդականությունները և k բնական թիվը: Միացնել այդ երկու հաջորդականությունները երրորդ՝ $c_0,c_1,\ldots,c_{n+m-1}$ հաջորդականության մեջ այնպես, որ երկրորդ հաջորդականությունն ընկնի առաջինի k-րդ և $k+1$-րդ տարրերի միջև ($n,m,k>0$), եթե հնարավոր չէ, տալ համապատասխան հաղորդագրություն:
24. Տրված են բնական թվերի $a_0,\ldots,a_{n-1}$ և $b_0,\ldots,b_{n-1}$ ($n\geq 1$) հաջորդականությունները: Ստանալ $c_0,\ldots,c_{n-1}$ հաջորդականությունը, որտեղ $c_i=true$, եթե a_i-ի և b_i-ի ամենամեծ ընդհանուր բաժանարարը մեծ է 7-ից, և $c_i=false$՝ հակառակ դեպքում:
25. Տրված են իրական թվերի $a_0,\ldots,a_{n-1}$ և $b_0,\ldots,b_{n-1}$ ($n\geq 1$) հաջորդականությունները և r իրական թիվը: Ստանալ $c_0,\ldots,c_{n-1}$ հաջորդականությունը, որտեղ $c_i=true$, եթե հար-

թության (a_i,b_i) կետը պատկանում է $(0,0)$ կենտրոնով և r շառավղով շրջանին և c_i=`false`՝ հակառակ դեպքում:

26. Տրված է հարթության կետերի (x_0,y_0), (x_1,y_1), … , (x_{n-1},y_{n-1}) հաջորդականությունը $(n \geq 1)$: Արտածել
    ա) կոորդինատների սկզբնակետից ամենահեռու ընկած կետի կոորդինատները,
    բ) այն երկու կետերի կարգահամարները, որոնց միջև հեռավորությունն ամենամեծն է,
    գ) այն կետի կոորդինատները, որից դեպի իրենից ամենահեռու ընկած կետի միջև եղած հեռավորությունը նվազագույնն է,
    դ) բոլոր այդ կետերն ընդգրկող և նվազագույն մակերեսով ուղղանկյան կողմերի երկարությունները:

27. Տրված է շրջանագծերի (x_0,y_0,r_0), (x_1,y_1,r_1), … , $(x_{n-1},y_{n-1},r_{n-1})$ հաջորդականությունը, որտեղ (x_i,y_i)-ն i-րդ շրջանագծի կենտրոնի կոորդինատներն են, իսկ r_i-ն նրա շառավիղն է: Արտածել
    ա) `YES`, եթե բոլոր շրջանագծերը ներդրված են մեկը մյուսի մեջ, և `NO`՝ հակառակ դեպքում,
    բ) `YES`, եթե շրջանագծերից ոչ մի երկուսը չունեն ընդհանուր մաս, և `NO`՝ հակառակ դեպքում,
    գ) այն շրջանագծի կարգահամարը, որը շոշափում է ամենաշատ քանակությամբ շրջանագծերի:

28. Տրված է բնական թվերի $a_0,\ldots,a_{n-1}$ հաջորդականությունը $(n \leq 32)$: Արտածել ամբողջ թիվ, որտեղ b_i=1 (b_i-ն ստացված թվի i-րդ բիթն է), եթե
    ա) a_i-ի 16-ական ներկայացումը պարունակում է `A` սիմվոլը,
    բ) a_i-ի i-րդ բիթը հավասար է 1-ի:
    Հակառակ դեպքում b_i=0:

29. Արտածել տրված $a_0,\dots,a_{n-1}$ և $b_0,\dots,b_{m-1}$ գործակիցներով բազմանդամների ($a_0+a_1x+\dots+a_{n-1}x^{n-1}$ և $b_0+b_1x+\dots+b_{m-1}x^{m-1}$) արտադրյալը (ստացված բազմանդամի գործակիցները):
30. Ներածել M բնական թիվը 2-ական տեսքով՝ ստանալով 0-ներից ու 1-երից կազմված զանգված և արտածել $M+1$ թվի 2-ական ներկայացումը:
31. Սահմանել 3 զանգված.
    `int integer1[20];int integer2[20];int sum[21];`
    Ներածել առավելագույնը 20 նիշ պարունակող երկու բնական թիվ և պահել դրանց թվանշանները առաջին երկու զանգվածներում: Հաշվել և արտածել դրանց գումարը՝ ստանալով արդյունքի թվանշանները `sum` զանգվածում:
32. Տրված են իրական թվերի $a_0,\dots,a_{n-1}$ և $b_0,\dots,b_{m-1}$ հաջորդականությունները: Կառուցել $c_0,c_1,\dots,c_{k-1}$ ($k\le n+m$) հաջորդականությունը, որը ստացվում է.
    ա) տրված հաջորդականությունների կցումից (այնպես, որ սկզբում լինի երկար հաջորդականությունը),
    բ) տրված հաջորդականությունների (որպես բազմություններ) միավորումից,
    գ) տրված հաջորդականությունների (որպես բազմություններ) հատումից:
33. Տրված են իրական թվերի $a_0,\dots,a_{n-1}$ և $b_0,\dots,b_{m-1}$ աճման կարգով դասավորված հաջորդականությունները: Առանց ներդրված ցիկլ օգտագործելու՝ կառուցել աճման կարգով դասավորված $c_0,c_1,\dots,c_{k-1}$ ($k\le n+m$) հաջորդականությունը, որը ստացվում է.
    ա) տրված հաջորդականությունների (որպես բազմություններ) միավորումից,
    բ) տրված հաջորդականությունների (որպես բազմություններ) հատումից:

6. Ֆունկցիաներ

Պրոցեդուրային ծրագրավորման էությունն այն է, որ բարդ խնդիրը լուծելու համար այն տրոհում ենք ենթախնդիրների, լուծում ենք այդ ենթախնդիրները՝ իրականացնելով համապատասխան ենթածրագրերը (պրոցեդուրաները), այնուհետև, օգտագործելով դրանք, լուծում ենք հիմնական խնդիրը: C++ լեզվում ենթածրագրերը կոչվում են ֆունկցիաներ: Ֆունկցիան ծրագրի հատված է, որը տրված մուտքային արժեքների համար կատարում է որոշակի գործողություններ և կարող է որպես արդյունք վերադարձնել մեկ արժեք: Ֆունկցիան օգտագործելու (կանչելու) համար պետք է այն հայտարարել և սահմանել: Ընդ որում՝ ֆունկցիայի հայտարարությունը կամ սահմանումը պետք է տրվի ֆունկցիայի կանչից առաջ: Ֆունկցիայի հայտարարությունը, որը կոչվում է նաև ֆունկցիայի նախատիպ, ունի հետևյալ տեսքը՝

տիպ անուն`(`*պարամետրեր*`);`

տիպ-ը ֆունկցիայի վերադարձվող արժեքի տիպն է, *անուն*-ը ֆունկցիայի անունն է, որը կարող է լինել ցանկացած նույնարկիչ, իսկ *պարամետրեր*-ը ֆունկցիայի մուտքային պարամետրերի ցուցակն է, որն ունի հետևյալ տեսքը՝

$տիպ_1$ *անուն*$_1$`,` … `,` $տիպ_n$ *անուն*$_n$

$n \geq 0$, $տիպ_i$-ն `i`-րդ պարամետրի տիպն է, իսկ $անուն_i$-ն `i`-րդ պարամետրի անունն է, որը կարող է նաև բացակայել, $i = 1, \ldots, n$: Եթե ֆունկցիան արժեք չի վերադարձնում, ապա վերադարձվող արժեքի տիպի փոխարեն գրվում է `void` ծառայողական բառը: Ֆունկցիայի սահմանումն ունի հետևյալ տեսքը՝

```
տիպ անուն(պարամետրեր) {
    մարմին
}
```

մարմին-ը ծրագրի հատված է, որը կատարվում է ֆունկցիան կանչելիս, իսկ մինչև բացվող ձևավոր փակագիծը կրկնվում է ֆունկցիայի նախատիպը: Ֆունկցիայի նախատիպում կամ սահմանման մեջ հանդիպող պարամետրերը կոչվում են ձևական (ֆորմալ) պարամետրեր: Ծրագրում ֆունկցիայի սահմանումը պետք է հանդիպի ճիշտ մեկ անգամ, իսկ նախատիպը կարող է հանդիպել անսահմանափակ քանակությամբ: Դա մասնավորապես հնարավորություն է տալիս ֆունկցիան կանչել տարբեր ֆայլերում՝ կանչից առաջ տալով միայն նախատիպը, իսկ ֆունկցիան սահմանել մեկ անգամ՝ ցանկացած ֆայլում: Այժմ լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է `a` և `b` բնական թվերը և վերադարձնում է դրանց ամենամեծ ընդհանուր բաժանարարը:

Լուծում՝ կներկայացնենք այս խնդրի լուծման երկու տարբերակ:

```
/*ամենամեծ ընդհանուր բաժանարարը հաշվում ենք Էվկլիդեսի  ալգորիթմի միջոցով՝ հանման եղանակով*/
unsigned int gcd1(unsigned int a,unsigned int b)
{
    //ենթադրում ենք, որ արգումենտները 0-ից մեծ են
    while(a!=b)
        if(a>b)
            a-=b;
        else
            b-=a;
    return a;
}
```

/*ամենամեծ ընդհանուր բաժանարարը հաշվում ենք Էվկլիդեսի ալգորիթմի միջոցով՝ բաժանման եղանակով, որն ավելի արդյունավետ է, քան լուծման նախորդ տարբերակը*/

```
unsigned int gcd2(unsigned int a,unsigned int b)
{
    while(b!=0)
    {
        unsigned int temp=b;
        b=a%b;
        a=temp;
    }
    return a;
}
```

Սահմանված ֆունկցիան աշխատեցնելու համար պետք է այն կանչել՝ մուտքային պարամետրերի փոխարեն փոխանցելով համապատասխան արժեքները: Ֆունկցիայի կանչն ունի հետևյալ տեսքը և այն կարելի է օգտագործել արտահայտություններում՝

անուն(*արժեք*$_1$, …, *արժեք*$_n$)

անուն-ը այն ֆունկցիայի անունն է, որը ցանկանում ենք կանչել, *արժեք*$_i$-ն ֆունկցիային փոխանցվող `i`-րդ արժեքն է, $i = 1, \ldots, n$, $n \geq 0$: Ֆունկցիայի կանչի ժամանակ փոխանցվող արժեքները կոչվում են նաև ֆունկցիայի փաստացի պարամետրեր: Փաստացի և ֆորմալ պարամետրերի քանակի և տիպերի համապատասխանությունը ստուգվում է կոմպիլյատորի կողմից և անհամապատասխանության դեպքում կոմպիլյացիան ավարտվում է սխալներով: Ընդ որում՝ ֆունկցիայի կանչի ժամանակ կոմպիլյատորը կարող է կատարել տիպերի անբացահայտ ձևափոխություն, որպեսզի համապատասխանեցնի փաստացի պարամետրի տիպը համապատասխան ֆորմալ պարամետրի

տիպին: Կանչենք նախորդ օրինակում իրականացված ֆունկցիաներից մեկը՝

```
//կանչվող ֆունկցիայի նախատիպը
unsigned int gcd2(unsigned int,unsigned int);
int main()
{
    unsigned int a,b;
    //մուտք ենք անում երկու ոչ բացասական ամբողջ թիվ
    cin>>a>>b;
    /*արտածում ենք մուտք արված թվերի ամենամեծ
    ընդհանուր բաժանարարը*/
    cout<<gcd2(a,b)<<endl;
    return 0;
}
```

C++ լեզուն հնարավորություն է տալիս ունենալ նույն անունով մեկից ավելի ֆունկցիաներ, որոնք տարբերվում են մուտքային պարամետրերի քանակով կամ տիպերով: Այդ դեպքում համապատասխան ֆունկցիան համարվում է վերաբեռնված: Եթե վերաբեռնված ֆունկցիայի կանչի ժամանակ կոմպիլյատորը չի կարողանում փաստացի պարամետրերի տիպերի հիման վրա միանշանակ հասկանալ, թե որ ֆունկցիան պետք է կանչել, ապա կոմպիլյացիան ավարտվում է սխալներով: Ստորև բերված է ֆունկցիայի վերաբեռնման օրինակ՝

```
//վերադարձնում է տրված երկու ամբողջ թվերից մեծագույնը
int max(int a, int b)
{
    return a>b?a:b;
}
//վերադարձնում է տրված երեք ամբողջ թվերից մեծագույնը
int max(int a, int b, int c)
{
    return a>b?(a>c?a:c):(b>c?b:c);
}
```

C++ լեզուն հնարավորություն է տալիս նաև ֆունկցիայի որոշ պարամետրերին տալ լռությամբ արժեք և կանչի ժամանակ չփոխանցել համապատասխան պարամետրերի արժեքները: Լռությամբ պարամետրերը պետք է տրված լինեն մինչև ֆունկցիայի կանչը հանդիպող նախատիպում կամ սահմանման մեջ: Տեղի ունի նաև հետևյալ սահմանափակումը՝ լռությամբ արժեք ունեցող որևէ պարամետրից հետո չի կարող հանդիպել պարամետր, որին լռությամբ արժեք տրված չէ: Դա վերաբերում է նաև ֆունկցիայի կանչին: Ստորև բերված է լռությամբ պարամետրեր ունեցող ֆունկցիայի օրինակ:

```
//ֆունկցիայի երկու պարամետրն էլ ունեն լռությամբ արժեք
void display(char c='*', int n=1);

int main()
{
    /* արտածում է *՝ երկու պարամետրի համար էլ
    օգտագործվում է լռությամբ արժեքները*/
    display();
    /* արտածում է +՝ երկրորդ պարամետրի համար
    օգտագործվում է լռությամբ արժեքը*/
    display('+');
    //արտածում է %%%
    display('%',3);
    return 0;
}

//տրված քանակությամբ արտածում է տրված սիմվոլը
void display(char c, int n)
{
    for(int i=1;i<=n;++i){
        cout<<c;
```

```
    }
    cout<<endl;
}
```

Ֆունկցիայի պարամետրերը կարող են փոխանցվել երկու հիմնական եղանակով՝

- Ըստ արժեքի: Այս դեպքում ֆունկցիային փոխանցվում են փաստացի պարամետրերի արժեքների կրկնօրինակները և ֆունկցիայի մարմնում պարամետրի արժեքը փոխելով հնարավոր չէ փոխել փաստացի պարամետրի արժեքը: Մինչև այժմ դիտարկված բոլոր օրինակներում պարամետրերը փոխանցվել են ըստ արժեքի:
- Հղումով: Այս դեպքում ֆունկցիային փոխանցվում են փաստացի պարամետրերը (իրենց հասցեների միջոցով) և ֆունկցիայի մարմնում պարամետրի արժեքի ցանկացած փոփոխության դեպքում փոխվում է նաև համապատասխան փաստացի պարամետրի արժեքը: Պարամետրը հղումով փոխանցելու համար փաստացի պարամետրի տիպից հետո ավելացնում ենք `&` սիմվոլը: Օրինակ՝

```
/*փոխում է տրված պարամետրերի արժեքները միմյանց հետ*/
void swap(int& a, int& b)
{
    int temp=a;
    a=b;
    b=temp;
}

int main()
{
    int a=10,b=20;
    //փոխում ենք տեղերով a-ի և b-ի արժեքները
    swap(a,b);
```

```
    //արտածում է 20,10
    cout<<a<<','<<b<<end;
    return 0;
}
```

Պարամետրերը հղումով փոխանցելու հնարավորությունը կարելի է օգտագործել նաև ծրագրի արդյունավետությունը բարձրացնելու նպատակով, քանի որ այդ դեպքում փոխանցվում է միայն փաստացի պարամետրի հասցեն և դրա արժեքի կրկնօրինակում տեղի չի ունենում՝ խնայելով ժամանակ և հիշողություն (մանավանդ մեծ չափի պարամետրերի դեպքում):

Ցուցիչ պարամետրերի կիրառման դեպքում տեղի է ունենում պարամետրի փոխանցում ըստ արժեքի. ցուցիչին հատկացված հիշողությունում պատճենվում է փաստացի պարամետրի արժեքը, որը հասցե է:

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն՝ ֆունկցիաների միջոցով:

1. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է `n` բնական թիվը և վերադարձնում է այդ թվի
   ա) ամենամեծ թվանշանի կարգահամարը (եթե այդ թվանշանը թվում հանդիպում է մի քանի անգամ վերադարձնել ամենաձախի կարգահամարը),
   բ) բոլոր պարզ բաժանարարների քանակը,
   գ) `16`-ական ներկայացման մեջ զրոների քանակը,
   դ) հարևան զույգ թվանշանների քանակը:
2. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է `n` բնական թիվը և վերադարձնում է
   ա) թե քանի անգամ է դրանում հանդիպում առաջին թվանշանը,

բ) դրանում կրկնվող թվանշանների միջին թվաբանականը,
գ) `true`, եթե դրանում թվանշանները դասավորված են չնվազման կարգով, և վերադարձնում է `false`՝ հակառակ դեպքում,
դ) `true`, եթե դրանում կան կրկնվող թվանշաններ, և վերադարձնում է `false`՝ հակառակ դեպքում:

3. Նկարագրել ֆունկցիա, որը վերադարձնում է արգումենտում ստացած `n>1` բնական թվին չգերազանցող
ա) ամենամեծ պարզ թիվը,
բ) պարզ թվերի քանակը,
գ) սիմետրիկ թվերի քանակը:
4. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է `a` և `b` բնական թվերը և վերադարձնում է
ա) դրանց ամենափոքր ընդհանուր բազմապատիկը,
բ) այդ թվերն իրար կցելուց ստացված թիվը (օրինակ՝ եթե `a=12` և `b=34`, ապա ֆունկցիան պետք է վերադարձնի `1234`):
5. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է `n` և `k` բնական թվերը և
ա) վերադարձնում է `true`, եթե `n`-ը հանդիսանում է `k`-ի որևէ ամբողջ աստիճան, և վերադարձնում է `false`՝ հակառակ դեպքում,
բ) արտածում է `n` թվի ներկայացումը `k`-ական համակարգում (`k<10`):
6. Նկարագրել ֆունկցիա, որն արգումենտում տրված `x` և `e` իրական թվերի համար վերադարձնում է հետևյալ արտահայտության արժեքը `e` ճշտությամբ՝
$x-x^3/3!+x^5/5!-x^7/7!+x^9/9!-x^{11}/11!+\ldots$

7. Գրել ֆունկցիա, որն արգումենտում տրված m և n ($m \le n$) բնական թվերի համար վերադարձնում է հետևյալ արտահայտության արժեքը.

$$\sum_{k=0}^{m} C_{n+k}^{2k}$$

8. Տրված բնական թվերի հաջորդականության համար արտածել
   ա) բոլոր 3-ով սկսվող թվերի գումարը (տրված թվի 3-ով սկսվելու ստուգումն իրականացնել ֆունկցիայի միջոցով),
   բ) դրանց ամենափոքր ընդհանուր բազմապատիկը (երկու թվերի ամենափոքր ընդհանուր բազմապատիկի հաշվարկն իրականացնել ֆունկցիայի միջոցով),
   գ) բոլոր այն թվերը, որոնք իրենց հարևանների միջին թվաբանականն են (տրված թվի՝ երկու այլ թվերի միջին թվաբանականը լինելու ստուգումն իրականացնել ֆունկցիայի միջոցով),
   դ) բոլոր այն թվերը, որոնք իրենց աջ հարևանի (վերջինի դեպքում առաջին տարրի) բնական աստիճանն են (երկու թվերի մեկը մյուսի աստիճան լինելու ստուգումն իրականացնել ֆունկցիայի միջոցով),
   ե) YES, եթե դրանում կա գոնե երկու սիմետրիկ թիվ, և արտածել NO՝ հակառակ դեպքում (տրված թվի սիմետրիկության ստուգումն իրականացնել ֆունկցիայի միջոցով),
   զ) այն տարրերի քանակը, որոնք առանց մնացորդի բաժանվում են իրենց բոլոր թվանշանների վրա (նշված հատկության ստուգումն իրականացնել ֆունկցիայի միջոցով)
9. Տրված բնական թվերի հաջորդականության համար հակառակ հերթականությամբ արտածել այն թվերը,
   ա) որոնք փոխադարձաբար պարզ են իրենց կարգահամարի հետ (երկու թվերի փոխադարձաբար պարզ լինելու ստու-

գումն իրականացնել ֆունկցիայի միջոցով),

բ) որոնց 3-ական ներկայացման մեջ 2 թվանշանը հանդիպում է զույգ անգամ (տրված թվի 3-ական ներկայացման մեջ 2 թվանշանի զույգ անգամ հանդիպելու ստուգումն իրականացնել ֆունկցիայի միջոցով),

գ) որոնք որևէ բնական թվի քառակուսին են (տրված թվի բնական թվի քառակուսի լինելու ստուգումն իրականացնել ֆունկցիայի միջոցով),

դ) որոնք 5-ի որևէ բնական աստիճանն են (տրված թվի 5-ի բնական աստիճան լինելու ստուգումն իրականացնել ֆունկցիայի միջոցով):

10. Արտածել տրված բնական թվերի հաջորդականությունում առաջին բացասական թվին նախորդող պարզ թվերի քանակը (եթե հաջորդականությունը բացասական թիվ չի պարունակում, արտածել `-1`): Թվի պարզության ստուգումն իրականացնել ֆունկցիայի միջոցով:

Պարամետրի փոխանցումը հղումով և ցուցիչ տիպի պարամետրեր

11. Նկարագրել ֆունկցիա, որը հղումով ստանում է օրը, ամիսը, տարին և նույն պարամետրերի միջոցով վերադարձնում է

ա) դրան նախորդող օրը,

բ) դրան հաջորդող `k` օր հետո ամսաթիվը (`k`-ն փոխանցվում է ֆունկցիային պարամետրի միջոցով),

գ) դրան նախորդող `k` օր առաջ ամսաթիվը (`k`-ն փոխանցվում է ֆունկցիային պարամետրի միջոցով):

12. Նկարագրել ֆունկցիա, որը ցուցիչ տիպի պարամետրերի միջոցով վերադարձնում է արգումենտում ստացված երկու իրական թվերի գումարը, տարբերությունը և արտադրյալը:

13. Նկարագրել ֆունկցիա, որը հղումով փոխանցված պարամետրերի միջոցով վերադարձնում է արգումենտում ստացված ամբողջ թվի մեծագույն և փոքրագույն թվանշանները:
14. Նկարագրել բուլյան ֆունկցիա, որն արգումենտում ստանում է երեք իրական թիվ և վերադարձնում է `false`, եթե գոյություն չունի տրված կողմերով եռանկյուն: Հակառակ դեպքում վերադարձնում է `true` և ցուցիչ տիպի պարամետրի միջոցով վերադարձնում է նաև այդ եռանկյան մակերեսը:

Զանգվածը որպես ֆունկցիայի պարամետր

15. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է ամբողջ թվերի հաջորդականություն և վերադարձնում է հաջորդականության
    ա) տարրերի միջին թվաբանականը,
    բ) ամենաշատ կրկնվող տարրը,
    գ) այն տարրերի քանակը, որոնք առանց մնացորդի բաժանվում են իրենց կարգահամարի վրա,
    դ) առաջին բացասական տարրի կարգահամարը կամ `-1`, եթե այն բացասական տարր չի պարունակում,
    ե) վերջին `0`-ի կարգահամարը կամ `-1`, եթե այն `0`-ներ չի պարունակում:
16. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է իրական գործակիցներով բազմանդամի $a_0, \dots, a_{n-1}$ գործակիցներն ու `k` իրական թիվը և վերադարձնում է
    ա) $a_0+a_1x+\dots+a_{n-1}x^{n-1}$ բազմանդամի արժեքը `k` կետում,
    բ) `true`, եթե `k`-ն $a_0+a_1x+\dots+a_{n-1}x^{n-1}$ բազմանդամի արմատ է, և վերադարձնում է `false`՝ հակառակ դեպքում:
17. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է բնական թվերի զանգված և վերադարձնում է
    ա) այն տարրերի քանակը, որոնք կարելի է ներկայացնել n^k

տեսքով, որտեղ `n`-ը և `k`-ն բնական թվեր են և `k>1`,

բ) բոլոր տարրերի ամենամեծ ընդհանուր բաժանարարը,

գ) բոլոր տարրերի մեջ `5` թվանշանի մասնակցության քանակը,

դ) `true`, եթե զանգվածի մեծագույն տարրը սիմետրիկ է, և վերադարձնում է `false`՝ հակառակ դեպքում,

ե) այն տարրերի քանակը, որոնք պարունակում են `k` թվանշանը, որտեղ `k`-ն փոխանցվում է ֆունկցիային պարամետրի միջոցով,

զ) այն տարրերի քանակը, որոնց թվանշանները դասավորված են չնվազման կարգով:

18. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է բնական թվերի զանգված և կառուցում է նոր զանգված՝ կազմված սկզբնական զանգվածի բոլոր տարրերում `0`-ից `9` թվանշանների կրկնությունների քանակներից:

19. Նկարագրել ֆունկցիա, որը կառուցում է զանգված՝

ա) կազմված այն եռանիշ թվերից, որոնց տասնավորների նիշը `2`-ով մեծ է հարյուրավորների նիշից,

բ) կազմված քառանիշ երջանիկ թվերից, այսինքն այնպիսի թվերից, որոնց առաջին երկու նիշերի գումարը հավասար է վերջին երկու նիշերի գումարին,

գ) կազմված աճման կարգով դասավորված `1000`-ից փոքր 2^i3^k տեսքի թվերից, որտեղ `i`-ն և `k`-ն դրական ամբողջ թվեր են,

դ) կազմված առաջին `n` պարզ թվերից, որտեղ `n`-ը փոխանցվում է ֆունկցիային պարամետրի միջոցով:

20. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է բնական թվերի զանգված և՝

ա) վերադասավորում է զանգվածի տարրերը հակառակ հերթականությամբ,

բ) զանգվածից հեռացնում է բոլոր պարզ թվերը,

գ) զանգվածում բոլոր պարզ թվերը փոխարինում է `0`-ներով:

21. Նկարագրել ֆունկցիա, որն արգումենտում ստանալով իրական թվերի աճման կարգով դասավորված հաջորդականությունը և `x` իրական թիվը, ավելացնում է այդ թիվը հաջորդականությանը, այնպես, որ չխախտվի կարգավորվածությունը:

22. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է իրական թվերի զանգված և վերադարձնում է՝

ա) `true`, եթե զանգվածը պարունակում է կրկնվող տարրեր, և վերադարձնում է `false`՝ հակառակ դեպքում,

բ) այն տարրերի քանակը, որոնք զանգվածում հանդիպում են ճիշտ 1 անգամ:

23. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է բնական թվերի զանգված և վերադարձնում է՝

ա) `true`, եթե զանգվածը պարունակում է գոնե `1` պարզ թիվ, և վերադարձնում է `false`՝ հակառակ դեպքում,

բ) `true`, եթե զանգվածը պարունակում է գոնե `1` թիվ, որը փոխադարձաբար պարզ է մեկ այլ պարամետրով ֆունկցիային փոխանցված `x` բնական թվի հետ, և վերադարձնում է `false`՝ հակառակ դեպքում,

գ) զանգվածի այն անդամների քանակը, որոնց `p`-ական ներկայացման մեջ չկան կրկնվող թվանշաններ, որտեղ `p` բնական թիվը փոխանցվում է ֆունկցիային պարամետրի միջոցով:

24. Նկարագրել ֆունկցիա, որն արգումենտում ստանում է նույն երկարության երկու բնական թվերի զանգված և՝

ա) վերադարձնում է երկու զանգվածների համապատասխան տարրերի կիսագումարներից մեծագույնը,

բ) կառուցում է նույն երկարության երրորդ զանգված, որի տարրերը հավասար են առաջին երկու զանգվածների

համապատասխան տարրերի ամենամեծ ընդհանուր բաժանարարներին:

Ֆունկցիայի ցուցիչը որպես ֆունկցիայի պարամետր

25. Նկարագրել ֆունկցիա, որը տրված ամբողջ թվերի $a_0, a_1, \ldots, a_{n-1}$ հաջորդականության և `f(x,y)` ֆունկցիայի համար վերադարձնում է հետևյալ արտահայտության արժեքը՝ $\max_{0 \le i < j < n} f(a_i, a_j)$, որտեղ `f(x,y)`-ը ֆունկցիա է `int` տիպի պարամետրերից և վերադարձնում է `double` տիպի արժեք: Անհրաժեշտ է նաև `main`-ից կանչել նկարագրված ֆունկցիան՝ ներածված հաջորդականության և երկու թվերի միջին թվաբանականը հաշվող ֆունկցիայի համար:
26. Նկարագրել ֆունկցիա, որը տրված ամբողջ թվերի $a_0, a_1, \ldots, a_{n-1}$ հաջորդականության և `f(x,y,z)` ֆունկցիայի համար վերադարձնում է հետևյալ արտահայտության արժեքը՝ $\max_{0 \le i < j < n} f(a_i, a_j, a_i \cdot a_j)$, որտեղ `f(x,y,z)`-ը ֆունկցիա է `int` տիպի պարամետրերից և վերադարձնում է `double` տիպի արժեք: Անհրաժեշտ է նաև `main`-ից կանչել նկարագրված ֆունկցիան՝ ներածված թվերի հաջորդականության և տրված թվերի միջին երկրաչափականը հաշվող ֆունկցիայի համար:
27. Նկարագրել ֆունկցիա, որը տրված իրական թվերի $a_0, a_1, \ldots, a_{n-1}$ հաջորդականության և `f(x)` ֆունկցիայի համար վերադարձնում է հետևյալ արտահայտության արժեքը՝ $\sum_{i=0}^{n-1} (a_i + f(a_i^i)) / i!$, որտեղ `f(x)`-ը ֆունկցիա է `double` տիպի պարամետրից և վերադարձնում է `double` տիպի արժեք: Անհրաժեշտ է նաև `main`-ից կանչել նկարագրված ֆունկցիան՝ ներածված թվերի հաջորդականության և տրված թվի քառակուսին հաշվող ֆունկցիայի համար:

28. Նկարագրել ֆունկցիա, որը տրված իրական թվերի $a_0, a_1, \ldots, a_{n-1}$ հաջորդականության և `f(x)` ֆունկցիայի համար վերադարձնում է հետևյալ արտահայտության արժեքը՝ $\sum_{i=0}^{n-1} i! / (f^1(a_0) + f^2(a_1) + \cdots + f^{i+1}(a_i))$, որտեղ `f(x)`-ը ֆունկցիա է `double` տիպի պարամետրից և վերադարձնում է `double` տիպի արժեք: Անհրաժեշտ է նաև `main`-ից կանչել նկարագրված ֆունկցիան՝ ներածված թվերի հաջորդականության և տրված թվի խորանարդը հաշվող ֆունկցիայի համար:
29. Նկարագրել ֆունկցիա, որը տրված իրական թվերի $a_0, a_1, \ldots, a_{n-1}$ հաջորդականության և `f(x,y)` ֆունկցիայի համար վերադարձնում է հետևյալ արտահայտության արժեքը՝ $\sum_{i=0}^{n-2} f(a_i^{i+1}, (-1)^i / a_{i+1})$, որտեղ `f(x,y)`-ը ֆունկցիա է `double` տիպի պարամետրերից և վերադարձնում է `double` տիպի արժեք: Անհրաժեշտ է նաև `main`-ից կանչել նկարագրված ֆունկցիան՝ ներածված թվերի հաջորդականության և երկու թվերի մեծագույնը հաշվող ֆունկցիայի համար:
30. Նկարագրել ֆունկցիա, որը տրված բնական թվերի $a_0, a_1, \ldots, a_{n-1}$ հաջորդականության և `f(x,y)` ֆունկցիայի համար վերադարձնում է հետևյալ արտահայտության արժեքը՝ $\max_{0 \le i < j < n} \sum_{k=i}^{j} \left(f(a_i, a_k) \cdot f(a_k, a_j)\right)$, որտեղ `f(x,y)`-ը ֆունկցիա է `int` տիպի պարամետրերից և վերադարձնում է `int` տիպի արժեք: Անհրաժեշտ է նաև `main`-ից կանչել նկարագրված ֆունկցիան՝ ներածված թվերի հաջորդականության և երկու թվերի ամենամեծ ընդհանուր բաժանարարը հաշվող ֆունկցիայի համար:

7. Անդրադարձ ֆունկցիաներ

Եթե ֆունկցիան ուղղակի (իր մարմնից) կամ անուղղակի (այլ ֆունկցիայի միջոցով) կերպով կանչում է ինքն իրեն, ապա այդ ֆունկցիան կոչվում է անդրադարձ ֆունկցիա: Անդրադարձ ֆունկցիայի մարմնում իր կանչը (ուղղակի կամ անուղղակի) պետք է տեղի ունենա միայն որոշակի պայմանի դեպքում, այլապես ֆունկցիան անընդհատ կկանչի ինքն իրեն մինչև պահունակի լցվելը (ֆունկցիայի կանչը և պարամետրերի փոխանցումը տեղի է ունենում պահունակի օգտագործման միջոցով) և ծրագրի վթարային ավարտը: Այժմ անդրադարձ ֆունկցիայի միջոցով լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Գրել անդրադարձ ֆունկցիա, որն արգումենտում ստանում է `n` բնական թիվը և վերադարձնում է `n`-ի մեծագույն թվանշանը:

Լուծում՝

```
unsigned short maxDigit(unsigned int n)
{
    if(n<10)
    {
        //միանիշ թվի մեծագույն թվանշանը ինքն է
        return n;
    }
    //առանձնացնում ենք վերջին թվանշանը
    unsigned short lastDigit=n%10;
    /*հեռացնում ենք թվի վերջին թվանշանը և հաշվում
    ենք ստացված թվի մեծագույն թվանշանը*/
    unsigned short max=maxDigit(n/10);
```

```
    /*վերադարձնում ենք lastDigit և max
    փոփոխականներից մեծի արժեքը*/
    return max>lastDigit?max:lastDigit;
}
```

Իհարկե ցանկացած խնդիր կարելի է լուծել առանց անդրադարձ ֆունկցիա օգտագործելու, սակայն անդրադարձ ֆունկցիայի օգտագործումը որոշ դեպքերում կարող է շատ անգամ պարզեցնել խնդրի լուծումը: Դիտարկենք այդպիսի մի օրինակ՝

Օրինակ 2. (Հանոյի աշտարակը) Տրված են A, B, C տառերով նշված երեք ձող, որոնցից առաջինի վրա մեծից փոքր դասավորված են իրարից տարբեր չափսերի N հատ սկավառակ (N>0): Յուրաքանչյուր քայլում թույլատրվում է ձողերից մեկի վերևում գտնվող սկավառակը տեղափոխել մեկ այլ ձողի վրա: Ընդ որում՝ արգելվում է մեծ սկավառակը դնել փոքր սկավառակի վրա: Անհրաժեշտ է տրված N դրական ամբողջ թվի համար արտածել նվազագույն թվով քայլերի հաջորդականությունը, որոնք անհրաժեշտ է կատարել A ձողի վրա գտնվող բոլոր N սկավառակները C ձողի վրա տեղափոխելու համար: Օրինակ՝ N=3 դեպքում պետք է արտածել քայլերի հետևյալ հաջորդականությունը՝ A->C, A->B, C->B, A->C, B->A, B->C, A->C:

Լուծում՝

```
#include <iostream>
using namespace std;

void hanoy(int n,char a,char b,char c)
{
    if(n==1){
       //եթե մեկ սկավառակ է, տեղափոխենք a-ից c
```

```
        cout<<a<<" -> "<<c<<endl;
        return;
    }
    //տեղափոխենք n-1 հատ սկավառակ a ձողից b ձողը
    hanoy(n-1,a,c,b);
    //տեղափոխենք վերջին սկավառակը a-ից c
    cout<<a<<" -> "<<c<<endl;
    //b-ից n-1 սկավառակները տեղափոխենք c
    hanoy(n-1,b,a,c);
}

int main()
{
     int n; //սկավառակների քանակը
     cin>>n;
     hanoy(n,'A','B','C');
     return 0;
}
```

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն՝ անդրադարձ ֆունկցիաների միջոցով:

1. Ի՞նչ կարտածի հետևյալ ծրագիրը.

ա)
```
#include <iostream>
using  namespace std;
int F(int a,int b){
    if(b==0) return a;
    int res=F(b,a%b);
    cout<<a<<' '<<b<<' '<<res<<endl;
    return res;
}
int main(){
    cout<<F(14,(1<<3))<<endl;
    return 0;
}
```

բ)
```
#include <iostream>
using  namespace std;
int F(int n,int k){
    if(n<=0) return 0;
    int res=F(n/k,k);
    cout<<n<<' '<<res<<endl;
    return res*10+(n%k);
}
int main(){
    cout<<F(25,(1<<2))<<endl;
    return 0;
}
```
գ)
```
#include <iostream>
using  namespace std;
int F(int n){
    if(n<=0) return 0;
    int res=F(n>>1)+(n&1);
    cout<<n<<' '<<res<<endl;
    return res;
}
int main(){
    cout<<F(50)<<endl;
    return 0;
}
```
դ)
```
#include <iostream>
using  namespace std;
int F(int n,int k){
    if(n<=0 || k<=0 || n==k) return 1;
    int res1=F(n-1,k);
    int res2=F(n-1,k-1);
    cout<<n<<' '<<k<<' '<<res1+res2<<endl;
    return res1+res2;
}
int main(){
    cout<<F(5,2)<<endl;
```

```
    return 0;
}
```

ե)
```
#include <iostream>
using  namespace std;
int F(int n,int k){
    if(k==0) return 1;
    int res=F(n,k-1);
    cout<<n<<' '<<k<<' '<<res<<endl;
    return res*n+1;
}
int main(){
    cout<<F(2,(1<<2))<<endl;
    return 0;
}
```

2. Գրել անդրադարձ ֆունկցիա, որը հաշվում է տրված `n` թվի ֆակտորիալը։
3. Օգտագործելով հետևյալ անդրադարձ առնչությունը՝ գրել անդրադարձ ֆունկցիա, որը հաշվում է տրված `a` և `b` ամբողջ թվերի գումարը.

   `a+b=a`, եթե `b=0`

   `a+b=(a+1)+(b-1)`, եթե `b>0`

   `a+b=(a-1)+(b+1)`, եթե `b<0`
4. Գրել անդրադարձ ֆունկցիա, որն արտածում է տրված ոչ բացասական ամբողջ թվի երկուական ներկայացումը։
5. Գրել անդրադարձ ֆունկցիա, որն արգումենտում ստանում է `n` բնական թիվը և վերադարձնում է

   ա) `1`-ից `n` թվերի գումարը,

   բ) `n`-ի թվանշանների քանակը,

   գ) `n`-ի թվանշանների գումարը,

   ե) `n` թվի ներկայացումը `8`-ական համակարգում՝ որպես `10`-ական համակարգի թիվ,

   զ) Ֆիբոնաչիի հաջորդականության `n`-րդ անդամը,

է) բնական թիվ, որը ստացվում է n թիվը շրջելուց (օրինակ՝ եթե $n=9812$, ապա պետք է վերադարձնել 2189):

6. Գրել անդրադարձ ֆունկցիա, որը վերադարձնում է արգումենտում ստացված երկու բնական թվերի ամենամեծ ընդհանուր բաժանարարը:
7. Գրել անդրադարձ ֆունկցիա, որը տրված x իրական և n դրական ամբողջ թվի համար հաշվում է x^n-ը՝ օգտվելով հետևյալ անդրադարձ առընչությունից.
   $x^n=1$, եթե $n=1$,
   $x^n=x^{n/2}x^{n/2}$, եթե n-ը զույգ է,
   $x^n=xx^{n-1}$, եթե n-ը 1-ից մեծ կենտ թիվ է:
8. Գրել անդրադարձ ֆունկցիա, որը արտածում է արգումենտում ստացված ամբողջ թվերի հաջորդականությունը հակառակ հերթականությամբ:
9. Գրել անդրադարձ ֆունկցիա, որը ներածում է տրված քանակով ամբողջ թվեր և արտածում է այդ թվերը հակառակ հերթականությամբ:
10. Գրել անդրադարձ ֆունկցիա, որն արգումենտում ստանալով $n\geq 0$ ամբողջ թիվը և x իրական թիվը՝ վերադարձնում է Չեբիշևի n-րդ բազմանդամի արժեքը x կետում: Չեբիշևի $T_n(x)$ բազմանդամը սահմանվում է հետևյալ ձևով.
    $T_0(x)=1$, $T_1(x)=x$
    $T_n(x)=2xT_{n-1}(x)-T_{n-2}(x), n\geq 2$
11. Տրված են բնական թվերի $a_0,\ldots,a_{n-1}$ և $b_0,\ldots,b_{n-1}$ ($n\geq 1$) հաջորդականությունները: Ստանալ $c_0,\ldots,c_{n-1}$ հաջորդականությունը, որտեղ $c_i=C_{a_i}^{b_i}$, եթե $a_i\geq b_i$, և $c_i=C_{b_i}^{a_i}$՝ հակառակ դեպքում: C_x^y-ի հաշվարկն իրականացնել անդրադարձ ֆունկցիայի միջոցով՝ օգտագործելով հետևյալ անդրադարձ առնչությունը.

$$C_x^0 = 1$$

$$C_x^y = C_x^{y-1} \cdot \frac{x - y + 1}{y}, y \geq 1$$

12. Լուծել նախորդ խնդիրը հետևյալ պահանջով. բոլոր քայլերը արտածելու փոխարեն պետք է արտածել միայն տրված K-րդ քայլը ($0<K<2^N$): Օրինակ՝ N=3 և K=4 մուտքային տվյալների դեպքում պետք է արտածել A->C:
13. Տրված են N տարբեր թվեր և M թիվը: Հաշվել, թե քանի եղանակով կարելի է տրված N թվերի բազմությունից ընտրել այնպիսի ենթաբազմություն, որ այդ ենթաբազմության թվերի գումարը հավասար լինի M-ի:
14. Գրել անդրադարձ ֆունկցիա, որը հաշվում է, թե տրված N բնական թիվը քանի եղանակով կարելի է ներկայացնել իրարից տարբեր թվերի գումարի տեսքով:

8. Երկչափ զանգվածներ (մատրիցներ)

Երկչափ ստատիկ զանգվածը (ընդհանուր դեպքում բազմաչափ զանգվածները) հայտարարելիս պետք է նշել բոլոր չափողականություններն առանձին-առանձին՝ ինդեքսային փակագծերի մեջ: Ստատիկ զանգվածի բոլոր չափողականությունները պետք է լինեն կոմպիլյացիայի ընթացքում հաշվարկվող արժեքներ: Երկչափ ստատիկ զանգվածի նկարագրման ընդհանուր տեսքը հետևյալն է՝

տիպ անուն[*չափ1*][*չափ2*] *սկզբնարժեքավորում*

տիպ-ը զանգվածի տարրերի տիպն է, *անուն*-ը զանգվածի անունն է, որը ցանկացած նույնարկիչ է, *չափ1*-ը կոմպիլյացիայի ընթացքում հաշվարկվող ամբողջ տիպի արտահայտություն է, որի արժեքը մատրիցի տողերի քանակն է, *չափ2*-ը կոմպիլյացիայի ընթացքում հաշվարկվող ամբողջ տիպի արտահայտություն է, որի արժեքը մատրիցի սյուների քանակն է իսկ *սկզբնարժեքավորում*-ը օգտագործվում է զանգվածի տարրերը սկզբնարժեքավորելու համար և կարող է բացակայել:

սկզբնարժեքավորում-ը ունի հետևյալ տեսքը՝

= { *տարրերի արժեքների ցուցակ* }

կամ

= { { *տարրերի արժեքների ցուցակ*$_1$ }, … , { *տարրերի արժեքների ցուցակ*$_n$ } } $n \geq 1$

Ձևավոր փակագծերում նշվում է մատրիցի տարրերի սկզբնական արժեքները, որոնք բաժանվում են ստորակետով: Առաջին դեպքում տող առ տող տրվում են մատրիցի տարրերի արժեքները: Երկրորդ դեպքում՝ նույնպես, սակայն խմբավորված ըստ տողերի: Ընդ որում՝ եթե տրված արժեքների քանակը ավելի

քիչ է պահանջվող տարրերի քանակից, ապա մնացած տարրերը սկզբնարժեքավորվում են համապատասխան տիպի լռելյայն արժեքով (որը C++-ի ներդրված տիպերի դեպքում հավասար է 0-ի): Ստորև բերված են ստատիկ երկչափ զանգվածի նկարագրման մի քանի օրինակ՝

- /*10x12 չափի int տիպի մատրից, որի տարրերը սկզբնարժեքավորված չեն*/
  ```
  int A[10][12];
  ```
- /*4x3 չափի int տիպի մատրից, որի առաջին տողը սկզբնարժեքավորում ենք մեկերով, երկրորդ տողը՝ երկուսներով, երրորդ տողը՝ երեքներով, իսկ մնացածը՝ զրոներով*/
  ```
  const int n=3;
  int A[n+1][n]={1,1,1,2,2,2,3,3,3};
  ```
- /*4x4 չափի int տիպի մատրից, որի անկյունագծից ներքև ընկած հատվածը սկզբնարժեքավորում ենք մեկերով, իսկ մնացածը՝ զրոներով*/
  ```
  int A[4][4]={{1},{1,1},{1,1,1},{1,1,1,1}};
  ```
- /*1000x1000 չափի int տիպի մատրից, որի տարրերը սկզբնարժեքավորում ենք 0-ներով*/
  ```
  int A[1000][1000]={0};
  ```
- /*2x3 չափի int տիպի մատրից, որի տարրերը սկզբնարժեքավորում ենք 1-ից 6 թվերով*/
  ```
  int A[][3]={1,2,3,4,5,6};
  ```

Երկչափ զանգվածով տարրերին կարող ենք դիմել հետևյալ եղանակով՝

անուն[*ինդեքս1*][*ինդեքս2*]

անուն-ը զանգվածի անունն է, *ինդեքս1*-ը ամբողջ տիպի արտահայտությունն է, որի արժեքը զանգվածի այն տարրի տողի հերթական համարն է (համարակալումը սկսվում է 0-ից), որին ցանկանում ենք դիմել, իսկ *ինդեքս2*-ը ամբողջ տիպի արտահայտությունն է, որի արժեքը զանգվածի այն տարրի սյան հերթա-

կան համարն է (համարակալումը սկսվում է 0-ից), որին ցանկանում ենք դիմել: Բազմաչափ զանգվածի տարրերը հիշողության մեջ դասավորվում են ըստ ամենաառաջին ինդեքսի փոփոխման կարգի, մատրիցի դեպքում՝ տող առ տող: Այժմ, օգտագործելով ստատիկ երկչափ զանգված, լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Հեռացնել տրված իրական թվերի մատրիցի բոլոր այն տողերը, որոնց առաջին տարրը մնացածի միջին թվաբանականն է: Մատրիցը պետք է ներկայացնել ստատիկ երկչափ զանգվածի միջոցով:

Լուծում՝ խնդիրը լուծելու համար կկատարենք մատրիցի տողերի վերադասավորում այնպես, որ չհեռացվող բոլոր տողերը դասավորվեն մատրիցի սկզբից սկսած՝ պահպանելով իրենց փոխադարձ դասավորությունը, իսկ վերջում կարտածենք միայն մատրիցի չհեռացված հատվածը:

```
#include <iostream>

using namespace std;

const int m=6; //սյուների քանակը

/*պարզում է մատրիցի տրված ինդեքսով տողը պետք է
հեռացվի, թե ոչ*/
bool isRowRemovable(double matrix[][m],
                    int rowIndex);

int main()
{
    const int n=5; //տողերի քանակը
    double matrix[n][m];
    //մատրիցի ներածում
    for(int i=0;i<n;++i)
```

```
    for(int j=0;j<m;++j)
        cin>>matrix[i][j];
/*այս զանգվածում հիշում ենք, թե որ տողերն են
հեռացվելու*/
bool isRemovable[n];
for(int i=0;i<n;++i)
{
    isRemovable[i]=isRowRemovable(matrix,i);
}
/*այս փոփոխականում ստանալու ենք չհեռացված
տողերի քանակը*/
int count=0;
/*մատրիցի տողերը վերադասավորում ենք այնպես,
որ սկզբում լինեն չհեռացված տողերը*/
for(int i=0;i<n;++i)
{
    if(!isRemovable[i])
    {
        if(i!=count)
        {
            /*i-րդ տողը տեղափոխում ենք տվյալ
            պահին վերջին չհեռացված տողից
            հետո*/
            for(int j=0;j<m;++j)
                swap(matrix[i][j],
                     matrix[count][j]);
            swap(isRemovable[i],
                 isRemovable[count]);
        }
        ++count;
    }
}
//արտածում ենք արդյունքը
```

```
    for(int i=0;i<count;++i)
    {
        for(int j=0;j<m;++j)
            cout<<matrix[i][j]<<' ';
        cout<<endl;
    }
    return 0;
}

bool isRowRemovable(double matrix[][m],
                    int rowIndex)
{
    double average=0;
    for(int i=0;i<m;++i)
    {
        average+=matrix[rowIndex][i];
    }
    //տրված տողի տարրերի միջին թվաբանականը
    average/=m;
    return average==matrix[rowIndex][0];
}
```

Երկչափ դինամիկ զանգված ստեղծելու համար նախ պետք է ստեղծենք ցուցիչների միաչափ դինամիկ զանգված, որի էլեմենտները լինելու են երկչափ զանգվածի տողերի հասցեները: Այնուհետև պետք է ստեղծել տողերը՝ որպես միաչափ դինամիկ զանգվածներ, և նրանց հասցեները հիշել ցուցիչների զանգվածում: Օրինակ, ամբողջ թվերի դինամիկ երկչափ զանգված կարող է ստեղծվել հետևյալ ձևով՝

```
int n=10; //մատրիցի տողերի քանակը
int m=20; //մատրիցի սյուների քանակը
//ստեղծում ենք ցուցիչների դինամիկ զանգվածը
int** A=new int*[n];
for(int i=0;i<n;++i){
```

```
        /*ստեղծում ենք i-րդ տողը ներկայացնող դինամիկ
        զանգվածը և դրա հասցեն վերագրում ենք ցուցիչների
        զանգվածի i ինդեքսով տարրին*/
        A[i]=new int[m];
}
```

Դինամիկ երկչափ զանգվածի տարրերին դիմում ենք ճիշտ նույն ձևով, ինչպես որ դիմում ենք ստատիկ երկչափ զանգվածի տարրերին: Հասկանալի է, որ դինամիկ երկչափ զանգվածն օգտագործելուց հետո համապատասխան միաչափ զանգվածները պետք է հեռացնել դինամիկ հիշողությունից: Վերևում բերված օրինակի համար դա կարող ենք անել հետևյալ ձևով՝

```
for(int i=0;i<n;++i){
     //հեռացնում ենք հիշությունից i-րդ տողը
     delete [] A[i];
}
//հեռացնում ենք հիշությունից ցուցիչների զանգվածը
delete [] A;
```

Այժմ լուծենք Օրինակ 1-ի խնդիրը՝ օգտագործելով դինամիկ երկչափ զանգված:

Օրինակ 2. Հեռացնել տրված իրական թվերի մատրիցի բոլոր այն տողերը, որոնց առաջին տարրը մնացածի միջին թվաբանականն է: Մատրիցը պետք է ներկայացնել դինամիկ երկչափ զանգվածի միջոցով:

Լուծում՝ խնդիրը լուծելու համար կկատարենք մատրիցի տողերի վերադասավորում այնպես, որ չհեռացվող բոլոր տողերը դասավորվեն մատրիցի սկզբից սկսած՝ պահպանելով իրենց փոխադարձ դասավորությունը, իսկ վերջում կարտածենք միայն մատրիցի չհեռացված հատվածը:

```
#include <iostream>
using namespace std;

//պարզում է տրված տողը պետք է հեռացվի, թե ոչ
bool isRowRemovable(double* row,int m);

int main()
{
    int n,m; //տողերի և սյուների քանակը
    cin>>n>>m;
    double** matrix=new double*[n];
    //մատրիցի տողերի ստեղծում և ներածում
    for(int i=0;i<n;++i)
    {
        matrix[i]=new double[m];
            for(int j=0;j<m;++j)
                cin>>matrix[i][j];
    }
    /*այս զանգվածում հիշում ենք, թե որ տողերն են
    հեռացվելու*/
    bool* isRemovable=new bool[n];
    for(int i=0;i<n;++i)
    {
        isRemovable[i]=
                    isRowRemovable(matrix[i],m);
    }
    /*այս փոփոխականում ստանալու ենք չհեռացված
    տողերի քանակը*/
    int count=0;
    /*մատրիցի տողերը վերադասավորում ենք այնպես,
    որ սկզբում լինեն չհեռացված տողերը*/
    for(int i=0;i<n;++i)
    {
```

```
        if(!isRemovable[i])
        {
            if(i!=count)
            {
                /*i-րդ տողը տեղափոխում ենք տվյալ
                պահին վերջին չհեռացված տողից
                հետո*/
                swap(matrix[i],matrix[count]);
                swap(isRemovable[i],
                     isRemovable[count]);
            }
            ++count;
        }
    }
    //արտածում ենք արդյունքը
    for(int i=0;i<count;++i)
    {
        for(int j=0;j<m;++j)
            cout<<matrix[i][j]<<' ';
        cout<<endl;
    }
    //հեռացնում ենք վերցված դինամիկ հիշողությունը
    for(int i=0;i<n;++i)
    {
        delete[] matrix[i];
    }
    delete[] isRemovable;
    delete[] matrix;
    return 0;
}

bool isRowRemovable(double* row,int m)
{
    double average=0;
```

```
    for(int i=0;i<m;++i)
    {
        average+=row[i];
    }
    //տրված տողի տարրերի միջին թվաբանականը
    average/=m;
    return average==row[0];
}
```

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն՝ օգտագործելով ստատիկ կամ դինամիկ երկչափ զանգված:

1. Տրված է բնական թվերի մատրից: Արտածել մատրիցի
   ա) այն տողերի քանակը, որոնցում կա գոնե երկու կենտ թիվ,
   բ) այն տողերի քանակը, որոնց բոլոր տարրերը՝ սկսած երկրորդից, բաժանվում են առաջին տարրի վրա,
   գ) այն տողերի քանակը, որոնցում կա գոնե մեկ կատարյալ թիվ,
   դ) այն տողերի տարրերի գումարները, որոնցում կա 5 թվանշանը պարունակող գոնե մեկ տարր,
   ե) այն տողերի տարրերի գումարները, որոնցում կա գոնե երկու պարզ թիվ,
   զ) այն տողերը, որոնց տարրերի գումարը պարզ թիվ է:
2. Տրված է իրական թվերի մատրից: Արտածել մատրիցի
   ա) բոլոր տարրերի գումարը,
   բ) բոլոր տողերի տարրերի գումարները,
   գ) այն տողերի քանակը, որոնց բոլոր տարրերը մեծ են 5-ից,
   դ) այն տողերի քանակը, որոնցում չկան կրկնվող տարրեր,
   ե) այն տողերի տարրերի գումարները, որոնցում բոլոր տարրերը մեծ են 5-ից,
   զ) բոլոր տողերի նվազագույն տարրերից մեծագույնը:

3. Տրված է բնական թվերի մատրից: Կառուցել և արտածել միաչափ զանգված,
   ա) որը պարունակում է մատրիցի բոլոր այն տողերի կարգահամարները, որոնցում կա 2-ի աստիճան հանդիսացող գոնե մեկ թիվ,
   բ) որի i-րդ տարրը հավասար է true, եթե մատրիցի i-րդ տողի պարզ թվերը կարգավորված են նվազման կարգով, և հավասար է false՝ հակառակ դեպքում,
   գ) որի i-րդ տարրը հավասար է մատրիցի i-րդ տողի այն թվերի միջին թվաբանականին, որոնց հազարավորների թվանշանը հավասար է միավորների թվանշանին,
   դ) որի i-րդ տարրը հավասար է true, եթե մատրիցի i-րդ սյան մեջ կա գոնե մեկ թիվ, որի 16-ական ներկայացումը պարունակում է E սիմվոլը, և հավասար է false՝ հակառակ դեպքում:
4. Տրված է ամբողջ թվերի մատրից: Կառուցել և արտածել միաչափ զանգված, որի
   ա) i-րդ տարրը հավասար է մատրիցի i-րդ տողի բացասական տարրերի քանակին,
   բ) i-րդ տարրը հավասար է մատրիցի i-րդ տողի դրական զույգ տարրերի գումարին (եթե այդպիսի թիվ չկա, ապա i-րդ տարրը հավասար է 0),
   գ) i-րդ տարրը հավասար է մատրիցի i-րդ տողի զրո չպարունակող թվերի արտադրյալին (եթե այդպիսի թիվ չկա, ապա i-րդ տարրը հավասար է 0),
   դ) i-րդ տարրը հավասար է 1-ի, եթե մատրիցի i-րդ տողի բոլոր թվերը պարունակում են 3 թվանշանը, և հավասար է −1՝ հակառակ դեպքում,
   ե) i-րդ տարրը հավասար է մատրիցի i-րդ սյան տարրերի միջին թվաբանականին:

5. Տրված է ամբողջ թվերի մատրից: Կարգավորել մատրիցի տողերը դրանցում
   ա) դրական տարրերի գումարների աճման կարգով,
   բ) բացասական տարրերի քանակի նվազման կարգով:
6. Տրված է բնական թվերի մատրից: Կարգավորել մատրիցի տողերը դրանցում
   ա) պարզ թվերի քանակների նվազման կարգով,
   բ) այնպիսի տարրերի քանակի աճման կարգով, որոնց առաջին բայթի 1-երի քանակը հավասար է վերջին բայթի 1-երի քանակին:
7. Տրված է իրական թվերի մատրից և x իրական թիվը: Մատրիցի i-րդ տողի բնութագիր կհամարենք հետևյալ արտահայտության արժեքը՝ $x^n+a_{i0}x^{n-1}+a_{i1}x^{n-2}+\ldots+a_{i(n-2)}x+a_{i(n-1)}$, որտեղ $a_{i0}, a_{i1}, \ldots, a_{i(n-1)}$-ը մատրիցի i-րդ տողի տարրերն են: Կարգավորել մատրիցի տողերը դրանց բնութագրերի աճման կարգով:
8. Հեռացնել տրված իրական թվերի մատրիցի բոլոր այն տողերը,
   ա) որոնց տարրերը համընկնում են առաջին տողի տարրերի հետ,
   բ) որոնց տարրերի գումարը փոքր է այդ տողի կարգահամարից,
   գ) որոնց տարրերը դասավորված են չնվազման կարգով,
   դ) որոնց տարրերի գումարը մեծագույնն է,
   ե) որոնք նշանափոխ հաջորդականություն են,
   զ) որոնք պարունակում են մատրիցի նվազագույն տարրին հավասար տարրեր:
9. Հեռացնել տրված բնական թվերի մատրիցի բոլոր այն տողերը, որոնց
   ա) վերջին տարրը մեծագույնն է տողում,

բ) տարրերը փոխադարձաբար պարզ են սյան կարգահամարի հետ (բացի 0 կարգահամարով տարրից),

գ) տարրերի գումարը փոխադարձաբար պարզ է այդ տողի կարգահամարի հետ:

10. Հեռացնել տրված իրական թվերի մատրիցի առաջին մեծագույն տարրը պարունակող տողը և սյունը:
11. Հեռացնել տրված իրական թվերի մատրիցի բոլոր զրոյական տողերը և սյուները: Արտածել առաջին այն տողի կարգահամարը, որը պարունակում է գոնե մեկ դրական թիվ (եթե այդպիսի տող չկա , արտածել −1):
12. Տրված իրական թվերի մատրիցի սկզբից ավելացնել մի նոր տող՝ բաղկացած սկզբնական մատրիցի համապատասխան սյուների նվազագույն տարրերից:
13. Տրված իրական թվերի մատրիցի վերջից ավելացնել մի նոր տող՝ բաղկացած սկզբնական մատրիցի համապատասխան սյուների

ա) տարրերի միջին թվաբանականներից,

բ) դրական տարրերի քանակներից:

14. Տրված իրական թվերի մատրիցի յուրաքանչյուր տողից հետո ավելացնել մի նոր տող՝ բաղկացած նախորդ տողի տարրերի քառակուսիներից:
15. Տրված իրական թվերի մատրիցում ավելացնել նոր տող այն առաջին տողից հետո, որում պահված է մատրիցի մեծագույն տարրը: Ավելացվող տողի տարրերը հավասար են այդ մեծագույնին:
16. Մատրիցի տարրն անվանենք թամբային կետ, եթե այն իր սյունում փոքրագույնն է, իսկ տողում՝ մեծագույնը: Արտածել տրված իրական թվերի մատրիցի բոլոր թամբային կետերը և դրանց կարգահամարները:

17. Տեղափոխել տրված իրական թվերի մատրիցի սյուներն այնպես, որ նվազման կարգով կարգավորված սյուները հայտնվեն մատրիցի աջ մասում:
18. Տրված է իրական թվերի մատրից: Կառուցել և արտածել նույն չափերով նոր մատրից, որի տարրերը հավասար են իրենց բոլոր հարևանների գումարին:
19. Արտածել տրված `nxm` և `mxk` չափանի իրական թվերի մատրիցների արտադրյալը:
20. Արտածել տրված `nxm` չափանի իրական թվերի մատրիցի և `m` չափանի իրական թվերի վեկտորի արտադրյալը:
21. Անհրաժեշտ է կառուցել և արտածել `nxm` չափանի մատրից, որը ստացվում է `1`-ից մինչև $n \cdot m$ ամբողջ թվերը հետևյալ հերթականությամբ այդ մատրիցում գրելուց.
    ա) սկզբում թվերը վերևից ներքև գրել առաջին սյունում, այնուհետև ներքևից վերև գրել երկրորդ սյունում, որից հետո վերևից ներքև գրել երրորդ սյունում և այդպես շարունակ մինչև մատրիցի վերջին սյունը: Օրինակ՝ `n=4, m=5` դեպքում պետք է արտածել հետևյալ մատրիցը.

    ```
    1  8  9   16  17
    2  7  10  15  18
    3  6  11  14  19
    4  5  12  13  20
    ```

    բ) շարժվելով անկյունակագծային ուղղությամբ (սկսած վերևի ձախ վանդակից)՝ զույգ կարգահամարով անկյունագծերի համար դեպի ներքև, իսկ կենտ կարգահամարով անկյունագծերի համար դեպի վերև: Օրինակ՝ `n=4, m=5` դեպքում պետք է արտածել հետևյալ մատրիցը.

```
1  3   4   10  11
2  5   9   12  17
6  8   13  16  18
7  14  15  19  20
```

գ) օձաձև հերթականությամբ՝ սկսելով վերևի ձախ վանդակից: Օրինակ՝ `n=4`, `m=5` դեպքում պետք է արտածել հետևյալ մատրիցը.

```
1   2   3   4   5
14  15  16  17  6
13  20  19  18  7
12  11  10  9   8
```

22. Անհրաժեշտ է ներածել տրված `nxm` չափի տախտակի վրա `X/0` խաղի քայլերի հաջորդականություն: Ներածումը պետք է կատարել մինչև մրցակիցներից մեկի հաղթանակը (երբ տախտակի վրա լինում է նվազագույնը տրված քանակով միևնույն սիմվոլից բաղկացած հորիզոնական, ուղղահայաց կամ անկյունկագծային շղթա) կամ ոչ-ոքի գրանցվելը: Վերջում պետք է արտածել խաղի արդյունքը:
23. Տրված բնական թվերի մատրիցի համար արտածել
    ա) այդ մատրիցի միևնույն թվից բաղկացած ամենաերկար հորիզոնական կամ ուղղահայաց հատվածի երկարությունը և դիրքը,
    բ) այդ մատրիցի միևնույն թվից բաղկացած տիրույթների քանակը:

Քառակուսային մատրիցներ

24. Տրված բնական թվերի քառակուսային մատրիցի համար արտածել

ա) վերին եռանկյան զույգ տարրերի գումարը,

բ) ստորին եռանկյան 3-ից մեծ և 10-ից փոքր տարրերի միջին թվաբանականը,

գ) երկրորդական անկյունագծի այն առաջին տարրը, որի թվանշանների գումարը հավասար է ինչ-որ թվի ֆակտորիալի,

դ) գլխավոր անկյունագծի այն տարրերի գումարը, որոնց թվանշանների գումարը մեծ է 10-ից,

ե) գլխավոր անկյունագծից վերև գտնվող տարրերի ամենամեծ ընդհանուր բաժանարարը:

25. Տրված իրական թվերի քառակուսային մատրիցի համար արտածել

ա) գլխավոր անկյունագծի մեծագույն տարրը և նրա կարգահամարը,

բ) երկրորդական անկյունագծից ներքև յուրաքանչյուր տողի տարրերից նվազագույնները,

գ) գլխավոր անկյունագծին զուգահեռ անկյունագծերի տարրերի գումարներից մեծագույնը,

դ) երկրորդական անկյունագծին կից երկու անկյունագծերի տարրերից մեծագույնի արժեքը,

ե) գլխավոր անկյունագծի մեծագույն տարրը: Տեղափոխել այն պարունակող տողը վերջին տողի հետ:

26. Տրված բնական թվերի քառակուսային մատրիցի համար արտածել YES, եթե

ա) նրա բոլոր տողերի, սյուների, գլխավոր ու երկրոդական անկյունագծերի համար տարրերի գումարը զույգ է,

բ) այն մոգական քառակուսի է. նրա բոլոր տողերի, սյուների, գլխավոր ու երկրոդական անկյունագծերի համար տարրերի գումարը նույնն թիվն է,

գ) այն նորմալ մոգական քառակուսի է. մատրիցում գրված

են 1-ից մինչև n^2 բոլոր բնական թվերը և այն մոգական քառակուսի է, որտեղ n-ը մատրիցի տողերի քանակն է:

Հակառակ դեպքում արտածել NO:

27. Շրջել տրված իրական թվերի քառակուսային մատրիցը գլխավոր (երկրորդական) անկյունագծի նկատմամբ:
28. Դիտելով տրված իրական թվերի քառակուսային մատրիցի գլխավոր անկյունագծից վերև գտնվող յուրաքանչյուր տողի տարրերը որպես բազմանդամի գործակիցներ՝ արտածել այդ բազմանդամների արժեքները տրված x կետում (Հորների սխեմայով):
29. Տրված իրական թվերի քառակուսային մատրիցի համար կառուցել և արտածել միաչափ զանգված, որի i-րդ տարրը հավասար է true, եթե մատրիցի գլխավոր անկյունագծին զուգահեռ i-րդ անկյունագծի տարրերի գումարը մեծ է 500-ից, և հավասար է false՝ հակառակ դեպքում:
30. Ներածել n բնական թիվը և կառուցել nxn չափի քառակուսային մատրից, որն ունի հետևյալ տեսքը.
    ա) երկրորդական անկյունագծի տարրերը հավասար են 1-ի, դրանից վերև 0-ներ են, իսկ ներքևում՝ n-եր: Օրինակ՝ n=5 դեպքում պետք է արտածել հետևյալ մատրիցը.

```
0  0  0  0  1
0  0  0  1  5
0  0  1  5  5
0  1  5  5  5
1  5  5  5  5
```

բ)
$$\begin{matrix} 1 & 2 & 3 & \dots & (n-1) & n \\ 1 & 2^2 & 3^2 & \dots & (n-1)^2 & n^2 \\ 0 & 0 & 3^3 & \dots & (n-1)^3 & n^3 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & (n-1)^{n-1} & n^{n-1} \\ 0 & 0 & 0 & \dots & 0 & n^n \end{matrix}$$

31. Ներածել n բնական թիվը և x իրական թիվը: Կառուցել $n \times n$ չափի քառակուսային մատրից, որն ունի հետևյալ տեսքը.

$$\begin{matrix} 1 & 1 & \dots & 1 \\ x & x & \dots & x \\ \dots & \dots & \dots & \dots \\ x^{n-1} & x^{n-1} & \dots & x^{n-1} \end{matrix}$$

32. Ներածել n բնական թիվը և $x_1, x_2, \dots, x_n$ իրական թվերը: Կառուցել $n \times n$ չափի քառակուսային մատրից, որն ունի հետևյալ տեսքը.

$$\begin{matrix} x_1 & x_2 & \dots & x_n \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \dots & \dots & \dots & \dots \\ x_1^n & x_2^n & \dots & x_n^n \end{matrix}$$

9. Տողեր

Տողը սիմվոլների հաջորդականություն է: C++ լեզվում տողը պետք է ավարտվի զրոյական '\0' սիմվոլով: Սովորաբար տողը պահում են char տիպի զանգվածում: Զանգվածում տողի սկզբնարժեքավորումը կարելի է կատարել այնպես, ինչպես դա արվում է թվային զանգվածներում: Օրինակ՝

```
char string_array[80]={'H', 'e', 'l', 'l', 'o', '!',
                       '\0'};
```

Նույնը կարելի է անել նաև հաստատուն տողերի միջոցով՝

```
char string_array[80]="Hello!";
```

cin-ի միջոցով մուտքագրումը իրականացնելու դեպքում ներածումը շարունակվում է մինչև դատարկ սիմվոլի հանդիպումը: Դատարկ սիմվոլները ներածելու համար հաճախ օգտագործվում է getline մեթոդը: Օրինակ՝

```
const int N=80;
char s[N];
cin.getline(s, N);
```

Այստեղ s զանգվածը նախատեսված է տողը պահելու համար: N-ը ցույց է տալիս ներածվող սիմվոլների մաքսիմալ քանակը: Ավելի ճիշտ, եթե ներածվող սիմվոլների քանակը N-1-ից շատ է ապա զանգված կտարվեն միայն առաջին N-1 սիմվոլները: Հակառակ դեպքում ներածումը կշարունակվի մինչև նոր տողի անցնելու կոդի ներածումը, որին համապատասխանում է '\n' սիմվոլը: Երկու դեպքում էլ ներածվող սիմվոլներին հաջորդող տարրին վերագրվում է '\0' սիմոլը:

Տողի արտածումը կարելի է կատարել cout-ի միջոցով՝

```
cout<<s<<endl;
```

Տողերն օգտագործելով լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Տրված տողերի հաջորդականության համար արտածել դրանց ամենաերկար ընդհանուր նախածանցը:

Լուծում՝

```
#include <iostream>
using namespace std;

/*հաշվում է տրված երկու տողերի ամենաերկար ընդհանուր նախածանցի երկարությունը՝ դիտարկելով առավելագույնը maxLength քանակի սիմվոլ*/
int greatestCommonPrefixLength(const char* str1,
                               const char* str2,
                               int maxLength);

int main()
{
    const int n=10; //հաջորդականության չափը
    /*տողերից յուրաքանչյուրի առավելագույն
    երկարությունը*/
    const int maxLength=50;
    //տողերի հաջորդականության զանգվածը
    char a[n][maxLength+1];
    for(int i=0;i<n;++i)
    {
        //ներածում ենք տողերի հաջորդականությունը
        cin>>a[i];
    }
    /*այս փոփոխականի մեջ ստանալու ենք բոլորի
    ամենաերկար ընդհանուր նախածանցի երկարությունը
    (ենթադրում ենք, որ n>1)*/
    int length=greatestCommonPrefixLength(a[0],
```

```
                                    a[1],maxLength);
    for(int i=2;i<n;++i)
    {
        /*հաշվում ենք ընթացիկ ամենաերկար ընդհանուր
        նախածանցի և i-րդ տողի ամենաերկար
        ընդհանուր նախածանցը*/
        length=greatestCommonPrefixLength(a[0],
                                        a[i],length);
    }
    /*արտածում ենք բոլորի ամենաերկար ընդհանուր
    նախածանցը*/
    for(int i=0;i<length;++i)
    {
        cout<<a[0][i];
    }
    cout<<endl;
    return 0;
}

int greatestCommonPrefixLength(const char* str1,
                               const char* str2,
                               int maxLength)
{
    int i=0;
    /*երկու տողերի վրայով շարժվում ենք այնքան ժամա-
    նակ, քանի դեռ չենք դիտարկել maxLength-ից շատ
    սիմվոլ, որևէ մեկի վերջին դեռ չենք հասել և երկու
    տողերի համապատասխան սիմվոլներն իրար հավա-
    սար են*/
    for(;i<maxLength && str1[i] && str2[i]
                     && str1[i]==str2[i];++i);

    return i;
}
```

`cstring` գրադարանում կան տողերի հետ աշխատելու համար մի շարք օգտակար ֆունկցիաներ, որոնց առաջարկում ենք ծանոթանալ ինքնուրույն: Դրանցից են՝ `strlen`, `strcmp`, `strcat`, `strcpy`, `strncmp`, `strncat`, `strncpy`, `strchr` և `strtok` ֆունկցիաները: `C++` լեզվի ստանդարտ գրադարանում տողերի հետ աշխատելու համար նախատեսված է նաև `string` դասը:

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն:

1. Գրել ծրագիր, որն առաջարկում է ներածել անուն և ապա արտածում է `"Hello, x"`, որտեղ `x`-ը ներածված անունն է:
2. Տրված է մինչև `60` սիմվոլից բաղկացած տող: Արտածել տողի մեջ մտնող բոլոր փոքրատառերը:
3. Տրված է մինչև `60` սիմվոլ պարունակող փոքրատառերից բաղկացած տող, որն ավարտվում է կետով: Արտածել այդ տողը՝ բոլոր տառերը դարձնելով մեծատառ:
4. Տրված է մինչև `60` սիմվոլ պարունակող մեծատառերից բաղկացած տող: Արտածել `YES`, եթե այդ տառերը դասավորված են այբբենական կարգով և արտածել `NO`՝ հակառակ դեպքում:
5. Տրված է մինչև `200` սիմվոլ պարունակող փոքրատառերից բաղկացած տող: Այբբենական կարգով արտածել տողի մեջ մտնող տառերը:
6. Տրված է մինչև `40` սիմվոլ պարունակող տող: Արտածել այդ տողը՝ նախապես դրանում կատարելով հետևյալ փոփոխությունը.
   ա) բոլոր `abc`-երը փոխարինել `def`-ով,
   բ) հեռացնել առաջին `w` տառը,
   գ) հեռացնել բոլոր `th`-երը,
   դ) առաջին `x`-ը փոխարինել `ks`-ով,
   ե) բոլոր `q` տառերից հետո ավելացնել `u`,

զ) բոլոր `ph`-երը փոխարինել `f`-ով, իսկ բոլոր `ed`-երը `ing`-երով:

7. Տրված է տող, որն `1`-ից `30` բառերի հաջորդականություն է: Յուրաքանչյուր բառ բաղկացած է `1`-ից `5` տառերից, հարևան բառերն իրարից անջատվում են ստորակետով, իսկ վերջին բառից հետո դրված է կետ: Արտածել տողի
   ա) բոլոր բառերը՝ հակառակ հերթականությամբ,
   բ) այն բառերը, որոնք այբբենական կարգով ավելի մեծ են իրենց նախորդ բառից և ավելի փոքր են իրենց հաջորդ բառից,
   գ) այն բառերը, որոնք չեն կրկնվում,
   դ) այն բառերը, որոնք տողում հանդիպում են ճիշտ երկու անգամ,
   ե) բոլոր բառերը՝ առանց կրկնությունների, նշելով նաև, թե տվյալ բառը քանի անգամ է կրկնվում տրված տողում,
   զ) բոլոր բառերը՝ այբբենական հերթականությամբ:

8. Տրված է տող, որը `2`-ից `50` բառերի հաջորդականություն է: Յուրաքանչյուր բառ բաղկացած է `1`-ից `8` տառերից, հարևան բառերն իրարից անջատվում են առնվազն մեկ բացատով, իսկ վերջին տառից հետո դրված է կետ: Արտածել տողի այն բառերը,
   ա) որոնք տարբերվում են վերջին բառից և սիմետրիկ են,
   բ) որոնք տարբերվում են վերջին բառից և որոնցում իրենց առաջին տառը հանդիպում է մեկից ավել անգամ,
   գ) որոնք տարբերվում են վերջին բառից և որոնցում տառերը դասավորված են այբբենական կարգով,
   դ) որոնց երկարությունը մեծագույնն է,
   ե) որոնցում կրկնվող տառ չկա,
   զ) որոնցում յուրաքանչյուր տառ մասնակցում է առնվազն երկու անգամ,
   է) որոնցում մեկումեջ անգլերեն այբուբենի ձայնավոր է:

9. Գրել ծրագիր, որը ներածում է տողեր (մինչև դատարկ տող հանդիպելը) և արտածում է դրանք՝ փոխարինելով դրանցում իրար հաջորդող մեկից ավելի բացատները մեկ բացատով:
10. Տրված տողի համար արտածել իրար հաջորդող թվանշանների ամենաերկար հաջորդականությունը:
11. Տրված տողի համար կառուցել և արտածել մի նոր տող,
    ա) որտեղ բոլոր իրար հաջորդող միևնույն սիմվոլների փոխարեն թողնված է դրանցից մեկը միայն,
    բ) որից հեռացված են իրար հաջորդող միևնույն սիմվոլների բոլոր հաջորդականությունները,
    գ) որից հեռացված են անգլերենի այբուբենում իրար հաջորդող առնվազն `3` փոքրատառերի հաջորդականությունները (`abc`, `bcde`, `cdefg` և այլն),
    դ) որից հեռացված են բոլոր թվանշանները,
    ե) որտեղ բոլոր `a` տառերը փոխարինված են `ab`-ով:
12. Տրված է մի քանի տողանոց տեքստ՝ բաղկացած անգլերեն այբուբենի տառերից, կետադրական նշաններից ու բացատներից: Համարելով, որ բառերն իրարից բաժանվում են բացատներով ու կետադրական նշաններով, իսկ նախադասությունները վերջանում են կետով՝
    ա) արտածել տեքստի բառերի, նախադասությունների և տողերի քանակները,
    բ) արտածել տեքստը՝ բոլոր բառերի առաջին տառերը դարձնելով մեծատառ, իսկ մնացածը՝ փոքրատառ,
    գ) արտածել տեքստի ամենաերկար բառերից որևէ մեկը,
    դ) արտածել տեքստի բոլոր `n` տառանի բառերը, որտեղ `n`-ը տրված թիվ է,
    ե) արտածել տեքստի մեծատառով սկսվող և փոքրատառով վերջացող բառերը,
    զ) արտածել տեքստում ամենաշատը հանդիպող `3` բառերը,

է) արտածել տեքստի ամենաշատ փոքրատառ պարունակող բառերը,

ը) արտածել տեքստի բոլոր բառերը՝ այբբենական կարգով և առանց կրկնությունների,

թ) արտածել տեքստը՝ բոլոր հարևան բառերի միջև մեկից ավելի բացատների փոխարեն թողնելով դրանցից մեկը, բոլոր նախադասությունների առաջին տառերը դարձնելով մեծատառ, իսկ տեքստի մնացած տառերը՝ փոքրատառ:

13. Ներածել մինչև `7` սիմվոլ պարունակող տող, որը բնական թվի `16`-ական ներկայացումն է: Արտածել այդ թվի `10`-ական ներկայացումը:
14. Ներածել մինչև `32` սիմվոլից բաղկացած տող: Արտածել `YES`, եթե այն `C++` լեզվի նույնարկիչ է, և արտածել `NO`՝ հակառակ դեպքում:
15. Տրված է մինչև `80` սիմվոլից բաղկացած տեքստ: Արտածել `YES`, եթե այն `C++` լեզվով գրված թվային հաստատուն է (օրինակ՝ `5`, `5.12`, `5.2E+4`), և արտածել `NO`՝ հակառակ դեպքում:
16. Գրել `int any(char* s1,char* s2)` նախատիպով ֆունկցիա, որը վերադարձնում է `s1` տողում `s2` տողի սիմվոլներից որևէ մեկի հետ համընկնող առաջին սիմվոլի կարգահամարը կամ `-1`, եթե այդպիսի սիմվոլ չկա:
17. Գրել `int find(char* str,char* substr)` նախատիպով ֆունկցիա, որը վերադարձնում է `str` տողում `substr` տողի առաջին անգամ հանդիպելու դիրքը և վերադարձնում է `-1`, եթե `substr` տողը չի հանդիպում `str` տողում:
18. Գրել ծրագիր, որը ներածում է մինչև `100` սիմվոլից բաղկացած տող ու արտածում է `YES`, եթե այն դարձգիր է, և արտածում է `NO`՝ հակառակ դեպքում: Դարձգիր կոչվում է այն նախադասությունը, որը և՛ սկզբից, և՛ վերջից կարդացվում է նույնությամբ, եթե դիտարկում ենք նախադասության միայն տառերը՝ անտեսելով բացատները, կետադրական նշանները

և այլ տառ չհանդիսացող սիմվոլները (օրինակ՝ "Madam I'm Adam" և "մենակ կատակ կանեմ" տողերը հանդիսանում են դարձգիր):

19. Տրված է տող, որը թվերից, բացվող ու փակվող փակագծերից և գործողության նշաններից բաղկացած մաթեմատիկական արտահայտություն է (հնարավոր է փակագծերը սխալ դրված լինեն): Արտածել YES, եթե հնարավոր է փակագծերի նման դասավորություն և NO՝ հակառակ դեպքում:
20. Գրել ծրագիր, որը ներածում է n բնական թիվը և մի քանի տողանոց տեքստ ու արտածում է այդ տեքստը՝ յուրաքանչյուր տողում թողնելով n սիմվոլ և հավասարեցնելով այն աջ կողմից (անհրաժեշտ քանակով բացատներ ավելացնելու միջոցով):
21. Գրել ֆունկցիա, որն արտածում է [1,999999] միջակայքից տրված բնական թվի ներկայացումը բառերի միջոցով:
22. Արտածել տրված բնական թիվը՝ օգտագործելով միայն * և բացատ սիմվոլները: Յուրաքանչյուր թվանշան պետք է արտածել 9x5 ուղղանկյուն տիրույթում: Երկու թվանշանների միջև պետք է թողնել 9x1 չափի ազատ տարածություն: Օրինակ՝ 15-ի համար պետք է արտածել հետևյալ պատկերը՝

```
            *     *  *  *  *  *
         *  *     *
      *     *     *
   *        *     *
*           *     *  *  *  *  *
            *                 *
            *                 *
            *                 *
            *                 *
            *     *  *  *  *  *
```

23. Ներածել երկու տող և արտածել դրանց ամենաերկար ընդհանուր ենթատողը:

10. Դասեր

Օբյեկտ-կողմնորոշված ծրագրավորման գաղափարախոսությունը կարելի է դիտարկել որպես պրոցեդուրային ծրագրավորման զարգացման արդյունք: Այն թույլ է տալիս իրար հետ փոխկապակցված տվյալները և դրանք մշակող ենթածրագրերը միավորել մեկ ամբողջության՝ դասի մեջ: Նկարագրված դասը նոր տիպ է և հնարավորություն է տրվում ստեղծել այդ նոր տիպի փոփոխականներ, որոնք կոչվում են տվյալ դասի օբյեկտներ: C++ լեզուն բացառապես օբյեկտ-կողմնորոշված ծրագրավորման լեզու չէ, սակայն այն հնարավորություն է տալիս ծրագրավորել՝ օգտագործելով օբյեկտ-կողմնորոշված ծրագրավորման մոտեցումները: Դասի նկարագրության ընդհանուր տեսքը C++ լեզվում հետևյալն է՝

```
class անուն {
  դասի մարմին
};
```

անուն-ը նկարագրվող դասի անունն է, իսկ *դասի մարմին*-ը պարունակում է նկարագրվող դասի անդամները, որոնք կարող են լինել անդամ փոփոխականներ (դաշտեր) կամ անդամ ֆունկցիաներ (մեթոդներ): Նկարագրված դասի օբյեկտ ստեղծելու համար օգտագործվում է ճիշտ նույն մոտեցումները, ինչ որ C++ լեզվի ներդրված տիպի փոփոխական ստեղծելու համար: Օրինակ՝

դաս օբյեկտ;

տողով ստեղծում ենք *օբյեկտ* անունով և *դաս* տիպի նոր փոփոխական: Տվյալ օբյեկտի անդամներին դիմելու համար (դաշտերի դեպքում արժեքը օգտագործելու կամ փոփոխելու համար, իսկ մեթոդների դեպքում դրանք կանչելու համար) օգտագործվում են `.` և `->` գործողությունները՝

օբյեկտ`.`*անդամ*

հղում. *անդամ*
ցուցիչ->*անդամ*

օբյեկտ-ը հանդիսանում է դասի օբյեկտ, *հղում*-ը հանդիսանում է դասի օբյեկտի հղում, *ցուցիչ*-ը հանդիսանում է դասի օբյեկտի ցուցիչ, իսկ *անդամ*-ը հանդիսանում է տվյալ դասի անդամ: Այսինքն՝ երբ դասի անդամին դիմում ենք դասի օբյեկտի ցուցիչի միջոցով, ապա օգտագործում ենք -> գործողությունը, հակառակ դեպքում՝. գործողությունը: Օբյեկտ-կողմնորոշված ծրագրավորման կարևոր սկզբունքներից են ինկապսուլյացիան, ժառանգումը և պոլիմորֆիզմը: Ժառանգմանն ու պոլիմորֆիզմին կանդրադառնանք հաջորդ բաժնում, իսկ այժմ անդրադառնանք ինկապսուլյացիային: Բացի տվյալները և դրանց մշակող ենթածրագրերը մեկ ամբողջության մեջ միավորելուց՝ դասերը հնարավորություն են տալիս նաև սահմանափակելու դասի անդամների հասանելիությունը՝ թույլ չտալով դիմել դրանց այդ դասի մաս չհանդիսացող ենթածրագրերից: Դասի անդամների հասանելիությունը սահմանելու համար C++ լեզվում օգտագործվում են տեսանելիության երեք մակարդակներ՝ `public`, `protected` և `private`: `public` տեսանելիություն ունեցող անդամներին հնարավոր է դիմել ծրագրի ցանկացած մասից, `protected` տեսանելիություն ունեցող անդամներին հնարավոր է դիմել միայն այդ դասի կամ դրանից ժառանգված դասերի մեթոդներից, իսկ `private` տեսանելությունն ունեցող անդամներին հնարավոր է դիմել միայն այդ դասի մեթոդներից: Դասերի անդամների տեսանելության մակարդակը սահմանելու համար դասի նկարագրության մեջ ավելացնում ենք տեսանելության բնորոշիչներ, որոնք ունեն հետևյալ տեսքը՝

մակարդակ:
  դասի անդամներ

մակարդակ-ը վերը նշված երեք տեսանելիության մակարդակներից որևէ մեկն է, իսկ *դասի անդամներ*-ը դասի որոշակի անդամներ են, որոնք ունենում են նշված տեսանելիությունը: Դասի մարմնում տեսանելիության բնորոշիչները կարող են հանդիպել ցանկացած քանակությամբ, ընդ որում՝ դրանք կարող են նաև կրկնվել: Այն դեպքում, երբ դասի որոշ անդամներ նկարագրվում են առանց իրենց տեսանելիությունը նշելու, ապա այդ անդամներն ունենում են `private` տեսանելիություն: Ստորև բերված է դասի նկարագրության և օգտագործման պարզ օրինակ՝

```
class Rational
{
public:
    //արտածում է օբյեկտը
    void Print();
    //փոխում է համարիչը տրված արժեքով
    void SetNumerator(int p);
    //վերադարձնում է համարիչի արժեքը
    int GetNumerator();
    /*փոխում է հայտարարը տրված արժեքով, եթե տրված
    արժեքը զրո չէ*/
    void SetDenominator(int q);
    //վերադարձնում է հայտարարի արժեքը
    int GetDenominator();
private:
    int numerator;
    int denominator;
};

int main()
{
    Rational r; //ստեղծում է Rational տիպի օբյեկտ
```

```
    r.SetNumerator(10);
    r.SetDenominator(11); //r=10/11
    r.Print();
    /*հաջորդ տողում կա քերականական սխալ, քանի որ
    դասի private անդամներին կարելի է դիմել միայն
    դասի մեթոդներից*/
    r.numerator=5;
    return 0;
}
```

Ինկապսուլյացիայի սկզբունքի ճիշտ կիրառումը շատ կարևոր է և տալիս է հետևյալ առավելությունները՝

- Իրականացնել դասն այնպես, որ դասի օբյեկտները միշտ գտնվեն ճիշտ և անհակասական վիճակում: Դասի օբյեկտի ճիշտ և անհակասական վիճակում գտնվելը կախված է այդ օբյեկտի դաշտերի արժեքներից: Եթե դասն իրականացնողը դաշտերի համար սահմանում է `private` տեսանելիություն, ապա դաշտերը հնարավոր է լինում փոփոխել միայն դասի մեթոդներից՝ թույլ տալով դասն իրականացնողին հետևել, որ դաշտերի արժեքները մշտապես լինեն ճիշտ և անհակասական: Օրինակ, եթե `Rational` դասում `denominator` դաշտի տեսանելիությունը լիներ `public`, ապա այդ դասն օգտագործողը կարող էր `r.denominator=0;` վերագրումով `r` օբյեկտի վիճակը դարձնել սխալ (քանի որ ռացիոնալ թվի հայտարարը չի կարող լինել զրո):
- Իրականացնել դասն այնպես, որ օգտագործողները կախված չլինեն դասի իրականացման մանրամասներից: Եթե դասն իրականացնողը միայն իրականացմանը վերաբերող դաշտերի և մեթոդների համար սահմանում է `private` տեսանելիություն, ապա դրանց հետագա փոփոխությունը չի կարող ազդել դասն օգտագործողների վրա: Օրինակ, եթե `Rational`

դասում `denominator` դաշտի տեսանելիությունը լիներ `public`, ապա դասն օգտագործողը կարող էր դիմել այդ դաշտին և դասն իրականացնողի կողմից այդ դաշտի հետագա փոփոխության (օրինակ՝ անվան փոփոխության) դեպքում ստիպված էր լինելու իր կոդերը նույնպես փոխել (այն բոլոր հատվածներում, որտեղ դիմել էր `denominator` դաշտին):

Անհրաժեշտության դեպքում դասն իրականացնողը կարող է հնարավորություն տալ դիմելու `protected` և `private` անդամներին նաև այլ դասերից և ֆունկցիաներից: Դա անելու համար պետք է համապատասխան դասերը և ֆունկցիաները դասի մարմնում հայտարարել որպես բարեկամ (`friend`) դասեր ու ֆունկցիաներ: Օրինակ՝

```
class դաս1
{
    …
    friend class դաս2;
    friend ֆունկցիա;
    …
};
```

դաս1-ը և *դաս2*-ը դասերի անուններ են, իսկ *ֆունկցիա*-ն ֆունկցիայի նախատիպ է: Այս օրինակում *դաս1*-ը իրականացնողը թույլատրում է *ֆունկցիա* նախատիպը ունեցող ֆունկցիայից և *դաս2*-ի մեթոդներից դիմել *դաս1* դասի `protected` և `private` անդամներին:

Այժմ անդրադառնանք դասի մեթոդների իրականացմանը: C++ լեզուն տրամադրում է դասի մեթոդների իրականացման երկու տարբերակ՝

- Դասի նկարագրության հետ միասին: Օրինակ՝ `Rational` դասի `SetDenominator` մեթոդի իրականացումը դասի մարմնում կունենա հետևյալ տեսքը՝

```
class Rational
{
    …
    void SetDenominator(int q)
    {
        /*եթե տրված արժեքը զրո է, ապա հայտարարը
        չենք փոխում*/
        if(q!=0){
            denominator=q;
        }
    }
    …
};
```

- Դասի սահմանումից դուրս: Այս դեպքում մեթոդը կարելի է իրականացնել նաև այլ ֆայլում: Օրինակ՝ `Rational` դասի `SetDenominator` մեթոդի իրականացումը դասի սահմանումից դուրս կունենա հետևյալ տեսքը՝

```
void Rational::SetDenominator(int q)
{
    /*եթե տրված արժեքը զրո է, ապա հայտարարը չենք
    փոխում*/
    if(q!=0){
        denominator=q;
    }
}
```

Այս դեպքում կարևոր է մեթոդի անվանումից առաջ դասի անվանումը նշելը (որին պետք է հաջորդի `::`), այլապես տրված սահմանումը չի տարբերվի սովորական ֆունկցիայի սահմանումից և կոմպիլյատորը չի կարող հասկանալ, թե որ դասի մեթոդի սահմանումն է տրվում:

`this` ցուցիչ

Դասի ոչ ստատիկ (ստատիկ մեթոդներին կանդրադառնանք ներքևում) մեթոդները սովորական ֆունկցիաներից տարբերվում են նրանով, որ դասի մեթոդը միշտ կանչվում է այդ դասի որևէ օբյեկտի համար: Ստացվում է, որ դասի ոչ ստատիկ մեթոդները բացի մեթոդի պարամետրերից ստանում են նաև լրացուցիչ անբացահայտ պարամետր՝ դասի օբյեկտը, որի համար այդ մեթոդը կանչվել է: C++ լեզուն հնարավորություն է տալիս մեթոդի մարմնում դիմել վերը նշված ընթացիկ օբյեկտին՝ օգտագործելով `this` ցուցիչը, որը բնականաբար հնարավոր է օգտագործել միայն դասի ոչ ստատիկ մեթոդներում: Օրինակ՝

```
void Rational::SetDenominator(int q)
{
     /*եթե տրված արժեքը զրո է, ապա հայտարարը չենք
     փոխում*/
     if(q!=0){
          /*denominator դաշտին դիմելու համար կարելի է
          օգտագործել this->denominator կամ պարզապես
          denominator: Երկու դեպքում էլ դիմում ենք
          ընթացիկ օբյեկտի համապատասխան դաշտին:*/
          this->denominator=q;
     }
}
```

Հասկանալի է, որ այս օրինակում this ցուցիչի օգտագործումը պարտադիր չէ, սակայն կան դեպքեր, երբ դասի մեթոդը հնարավոր չէ իրականացնել առանց այդ ցուցիչն օգտագործելու: Դիտարկենք այդպիսի երկու օրինակ՝

- այն բոլոր դեպքերում, երբ մեթոդի որևէ պարամետրի կամ մեթոդի մարմնում օգտագործված լոկալ փոփոխականի անունը համընկնում է դաշտի անվան հետ, ապա ընթացիկ օբյեկտի այդ դաշտին դիմելու համար պետք է օգտագործել this ցուցիչը, այլապես կդիմենք նույնանուն պարամետրին կամ լոկալ փոփոխականին:
- այն դեպքում, երբ մեթոդը պետք է վերադարձնի ընթացիկ օբյեկտի հասցեն կամ հղումը, ապա կրկին պետք է օգտագործել this ցուցիչը՝

```
return this; //վերադարձնում է ընթացիկ օբյեկտի
             //հասցեն
return *this; //վերադարձնում է ընթացիկ օբյեկտի
              //հղումը
```

Կոնստրուկտոր և դեստրուկտոր

Կոնստրուկտորը դասի հատուկ մեթոդ է, որը կանչվում է դասի օբյեկտ ստեղծելիս՝ այն սկզբնարժեքավորելու նպատակով: Չի կարող ստեղծվել դասի որևէ օբյեկտ առանց կոնստրուկտոր կանչվելու: Դասը կարող է պարունակել մեկից ավելի կոնստրուկտոր, որոնք տարբերվում են իրարից պարամետրերով: Ի տարբերություն դասի սովորական մեթոդների, կոնստրուկտորներն ունեն հետևյալ առանձնահատկությունները՝

- Կոնստրուկտորի անունը պետք է համընկնի դասի անվան հետ: Մնացած մեթոդների դեպքում անունը չի կարող համընկնել դասի անվան հետ:

- Կոնստրուկտորը արժեք չի վերադարձնում և նկարագրության մեջ վերադարձվող արժեքի տիպ չի նշվում (նույնիսկ `void`): Ստորև բերված է `Rational` դասի կոնստրուկտորի նկարագրության և իրականացման օրինակ՝

```
class Rational
{
     …
     Rational(int numerator=0,
              int denominator=1)
     {
          this->numerator=numerator;
          /*չենք մոռանում բացառել հայտարարը 0-ով
          սկզբնարժեքավորելու դեպքը*/
          this->denominator=denominator!=0
                                  ?denominator:1;
     }
     …
};
```

- Կոնստրուկտորի կողմից դաշտերի սկզբնաժեքավորումը կատարվում է մինչև կոնստրուկտորի մարմնի կատարումը՝ սկզբնարժեքավորման ցուցակի միջոցով՝

կոնստրուկտոր (*պարամետրեր*)
    : *ցուցակ*
{
    մարմին
}

կոնստրուկտոր-ը դասի անվանումն է, *պարամետրեր*-ը կոնստրուկտորի ֆորմալ պարամետրերն են, *ցուցակ*-ը վե-

րևում նշված սկզբնարժեքավորման ցուցակն է, իսկ *մարմին*-ը կոնստրուկտորի մարմինն է:

Ընդհանուր դեպքում սկզբնարժեքավորման ցուցակն ունի հետևյալ տեսքը՝

$$\textit{դաշտ}_1(\textit{արժեքներ}_1), \textit{դաշտ}_2(\textit{արժեքներ}_2), \dots, \textit{դաշտ}_n(\textit{արժեքներ}_n), n \geq 1$$

$\textit{դաշտ}_i$-ն ($i = 1, \dots, n$) օբյեկտի դաշտի անունն է, որը ցանկանում ենք սկզբնարժեքավորել: Եթե $\textit{դաշտ}_i$-ն այլ դասի օբյեկտ չէ, ապա $\textit{արժեքներ}_i$-ն այն արժեքն է, որով դաշտը պետք է սկզբնարժեքավորվի, հակառակ դեպքում տեղի է ունենում համապատասխան դասի կոնստրուկտորի կանչ և $\textit{արժեքներ}_i$-ն այդ կանչի ժամանակ փոխանցվող փաստացի պարամետրերի ցուցակն է ($i = 1, \dots, n$):

Սկզբնարժեքավորման ցուցակում դաշտերը չեն կարող կրկնվել, իսկ դաշտերի հերթականությունը ոչ մի դեր չի խաղում: Որպես օրինակ ձևափոխենք `Rational` դասի վերևում նկարագրված կոնստրուկտորը՝

```
class Rational
{
     …
     Rational(int numerator=0,
              int denominator=1)
     :numerator(numerator),
      denominator(denominator!=0?denominator:1)
     { }
    …
};
```

Դասի օբյեկտ ստեղծելիս կատարվում է համապատասխան կոնստրուկտորի կանչ (գործում են ֆունկցիաների վերաբեռնման կանոնները)՝

```
/*կանչվում է այն կոնստրուկտորը, որը հնարավոր է կանչել
առանց պարամետրերի*/
դաս օբյեկտ;

/*կանչվում է այն կոնստրուկտորը, որը հնարավոր է
կանչել արժեքներ փաստացի պարամետրերով*/
դաս օբյեկտ(արժեքներ);
```

Դասի որոշ կոնստրուկտորներ ունեն հատուկ նշանակություն: Դրանք են՝ լռությամբ կոնստրուկտորը, պատճենման կոնստրուկտորը և տիպի ձևափոխության կոնստրուկտորը: Այն կոնստրուկտորը, որը հնարավոր է կանչել առանց պարամետրերի, կոչվում է լռությամբ կոնստրուկտոր: Վերևում նկարագրված `Rational` դասի կոնստրուկտորը նույնպես լռությամբ կոնստրուկտոր է: Լռությամբ կոնստրուկտորն ունի հետևյալ առանձնահատկությունները՝

- Այն դեպքում, երբ դասը չի պարունակում որևէ կոնստրուկտոր, կոմպիլյատորը փորձում է գեներացնել լռությամբ կոնստրուկտոր այդ դասի համար: Գեներացված լռությամբ կոնստրուկտորը ունենում է `public` տեսանելիություն և սկզբնարժեքավորման ցուցակի միջոցով կանչում է դասի օբյեկտ հանդիսացող դաշտերի լռությամբ կոնստրուկտորները: Եթե որևէ դաշտի համար դա չի հաջողվում (օրինակ, եթե համապատասխան դասը չունի լռությամբ կոնստրուկտոր կամ այն ունի `private` տեսանելիություն), ապա լռությամբ կոնստրուկտորը չի գեներացվում:
- Եթե դասի կոնստրուկտորի սկզբնարժեքավորման ցուցակից բացակայում է այդ դասի դաշտը, որը որևէ դասի օբյեկտ է, ապա կոմպիլյատորը փորձում է ավելացնել այդ դաշտը սկզբնարժեքավորման ցուցակի մեջ՝ նրա համար կանչելով

լռությամբ կոնստրուկտոր: Եթե դա չի հաջողվում (օրինակ նախորդ կետում նշված պատճառներով), ապա ծրագրի կոմպիլյացիան ավարտվում է համապատասխան քերականական սխալներով:

Այժմ անդրադառնանք պատճենման կոնստրուկտորին: Այն կոնստրուկտորը, որը կարող է կանչվել՝ որպես պարամետր ստանալով նույն տիպի մեկ այլ օբյեկտ, կոչվում է պատճենման կոնստրուկտոր: Անվանումը պայմանավորված է նրանով, որ այդ կոնստրուկտորը օգտագործվում է նոր օբյեկտ ստեղծելու համար, որը լինելու է գոյություն ունեցող օբյեկտի պատճեն: Պատճենման կոնստրուկտորը ունի հետևյալ նախատիպը՝

դաս(const *դաս*&);

դաս-ը համապատասխան դասի անունն է: Պատճենման կոնստրուկտորի պարամետրը պետք է հղումով փոխանցել, այլապես կոմպիլյատորը պետք է պատճենի փաստացի պարամետրը կոնստրուկտորը կանչելուց առաջ, որը կբերի անվերջ ռեկուրսիայի, քանի որ պատճենումը պետք է կատարվի այդ նույն կոնստրուկտորի միջոցով: Պարամետրի փոխանցումը որպես const պարամետր ցանկալի է, քանի որ այն հնարավորություն է տալիս պատճենել նաև հաստատուն օբյեկտները և ճիշտ իրականացման դեպքում կոնստրուկտորը միևնույնն է չպետք է փոփոխության ենթարկի փոխանցված օբյեկտը: Օրինակ՝ իրականացնենք պատճենման կոնստրուկտոր Rational դասի համար՝

```
class Rational
{
    ...
    Rational(const Rational& other)
        :numerator(other.numerator),
         denominator(other.denominator)
    { }
```

```
    …
};
```

Պատճենման կոնստրուկտորն ունի հետևյալ առանձնահատկությունները՝

- Այն դեպքում, երբ դասը չի պարունակում պատճենման կոնստրուկտոր, կոմպիլյատորը փորձում է գեներացնել պատճենման կոնստրուկտոր այդ դասի համար: Գեներացված պատճենման կոնստրուկտորը ունենում է `public` տեսանելիություն և սկզբնարժեքավորման ցուցակի միջոցով կանչում է դասի օբյեկտ հանդիսացող դաշտերի պատճենման կոնստրուկտորները՝ որպես պարամետր փոխանցելով պատճենվող օբյեկտի համապատասխան դաշտը, իսկ դասի օբյեկտ չհանդիսացող դաշտերը սկզբնարժեքավորում է պատճենվող օբյեկտի համապատասխան դաշտերի արժեքներով: Եթե որևէ դաշտի համար դա չի հաջողվում (օրինակ, եթե համապատասխան դասը չունի պատճենման կոնստրուկտոր կամ այն ունի `private` տեսանելիություն), ապա պատճենման կոնստրուկտոր չի գեներացվում:
- Պատճենման կոնստրուկտորը կարող է կանչվել ինչպես բացահայտ եղանակով, այնպես էլ կոմպիլյատորի կողմից անբացահայտ եղանակով՝ այն բոլոր դեպքերում, երբ անհրաժեշտ է լինում ստեղծել որևէ օբյեկտի կրկնօրինակ: Օրինակ՝ երբ ֆունկցիայի պարամետրը փոխանցվում է ըստ արժեքի, ապա ֆունկցիայի կանչի ժամանակ կոմպիլյատորը պետք է պատճենի փաստացի պարամետրը, որը դասի օբյեկտների համար իրականացվում է պատճենման կոնստրուկտորի միջոցով: Բացահայտ եղանակով պատճենման կոնստրուկտորի կանչի օրինակ է հետևյալը՝

```
Rational r1(10,11); //r1=10/11
```

```
/*երկու դեպքում էլ օբյեկտը ստեղծվում է պատճենման
կոնստրուկտորի միջոցով*/
Rational r2=r1;
Rational r3(r1);
```

Քանի որ պատճենման կոնստրուկտորը կարող է գեներացվել կոմպիլյատորի կողմից, ապա ոչ միշտ է իմաստ ունենում իրականացնել այն: Մասնավորապես՝ վերևում ներկայացված `Rational` դասի պատճենման կոնստրուկտորի իրականացումը համընկնում է կոմպիլյատորի կողմից գեներացված տարբերակի իրականացման հետ: Պատճենման կոնստրուկտորը կարիք է լինում իրականացնելու այն դեպքերում, երբ դաշտերի պարզագույն պատճենումը ցանկալի չէ: Մասնավորապես՝ այն դեպքում, երբ դասի դաշտերից մեկը դինամիկ զանգվածի ցուցիչ է, ապա կոմպիլյատորի կողմից իրականացման դեպքում պատճենումից հետո երկու օբյեկտների համար այդ դաշտերը ցույց կտան միևնույն զանգվածի վրա, այնինչ հիմնականում կարիք է լինում դինամիկ զանգվածը նույնպես պատճենել, որը հնարավոր է անել՝ պատճենման կոնստրուկտորի համապատասխան իրականացման միջոցով:

Անդրադառնանք տիպի ձևափոխության կոնստրուկտորին: Այն բոլոր դեպքերում, երբ `T` դասում ունենք կոնստրուկտոր, որը կարելի է կանչել `A` տիպի մեկ պարամետրով, կոնստրուկտորը կարող է այն օգտագործել `A` տիպից դեպի `T` տիպ բացահայտ կամ անբացահայտ ձևափոխության համար, քանի որ այդ կոնստրուկտորի միջոցով կարելի է ստեղծել `T` տիպի օբյեկտ՝ `A` տիպի արժեքի հիման վրա: Դա է պատճառը, որ այդպիսի կոնստրուկտորը կոչվում է նաև տիպի ձևափոխության կոնստրուկտոր: Եթե դասն իրականացնողը չի ցանկանում, որ տիպի ձևափոխության կոնստրուկտորը օգտագործվի տիպերի անբացահայտ ձևափո-

խության համար, ապա այդ կոնստրուկտորի նկարագրության դիմացից պետք է ավելացնել `explicit` ծառայողական բառը՝

`explicit` *կոնստրուկտոր;*

Տիպի ձևափոխության կոնստրուկտորի օրինակ է վերևում ներկայացված `Rational` դասի երկու `int` տիպի պարամետրով կոնստրուկտորը: Քանի որ այն կարելի է կանչել մեկ `int` տիպի պարամետրով (երկրորդ պարամետրն ունի լռությամբ արժեք), ապա այդ կոնստրուկտորը հնարավոր է օգտագործել `int` տիպից դեպի `Rational` տիպ բացահայտ և անբացահայտ ձևափոխություն իրականացնելու համար:

Դեստրուկտորը նույնպես դասի հատուկ մեթոդ է, որը կանչվում է, երբ ավարտվում է դասի օբյեկտի կյանքի տևողությունը և անհրաժեշտ է հեռացնել այն հիշողությունից: Ի տարբերություն կոնստրուկտորի, դեստրուկտորը միակն է և չունի պարամետրեր: Այն նույնպես արժեք չի վերադարձնում և նկարագրության մեջ վերադարձվող արժեքի տիպ չի նշվում (նույնիսկ `void`): Դասի դեստրուկտորի անունը ստացվում է դասի անվան դիմացից ~ սիմվոլը ավելացնելով: Ստորև բերված է `Rational` դասի դեստրուկտորի սահմանումը՝

```
class Rational
{
    ...
    ~Rational()
    {
        cout<<"Rational class destructor.\n";
    }
    ...
};
```

Դասում դեստրուկտորի առկայությունը պարտադիր չէ: Դեստրուկտորի իրականացումը դառնում է անհրաժեշտ մի շարք

դեպքերում: Մասնավորապես այն կարող է օգտագործվել օբյեկտի կյանքի ընթացքում զբաղեցրած (օրինակ՝ կոնստրուկտորի աշխատանքի ընթացքում) դինամիկ հիշողությունը կամ այլ ռեսուրսներ ազատելու համար:

Դասի `const` և `static` անդամներ

Եթե դասի ոչ ստատիկ դաշտի դիմացից գրված է `const` ծառայողական բառը, ապա այդ դաշտը դառնում է հաստատուն: Դասի հաստատուն դաշտը թույլատրվում է սկզբնարժեքավորել կոնստրուկտորի սկզբնարժեքավորման ցուցակի միջոցով, որից հետո արգելվում է փոփոխել այդ դաշտի արժեքը: Դասի ոչ ստատիկ մեթոդը նույնպես կարող է հայտարարվել որպես `const` մեթոդ (բացառությամբ կոնստրուկտորների և դեստրուկտորի)՝ պարամետրերի փակվող փակագծից հետո ավելացնելով `const` ծառայողական բառը: `const` մեթոդը մնացած մեթոդներից տարբերվում է նրանով, որ `const` մեթոդին արգելվում է փոփոխել ընթացիկ օբյեկտի դաշտերը: Այլ կերպ ասած՝ `const` մեթոդի ներսում `this` ցուցիչը դառնում է հաստատուն ցուցիչ: Որպես հետևանք՝ `const` մեթոդից չի թույլատրվում կանչել ոչ `const` մեթոդ ընթացիկ օբյեկտի համար: Հասկանալի է, որ դասի հաստատուն օբյեկտների համար թույլատրվում է կանչել դասի միայն `const` մեթոդները: Դա է պատճառը, որ ցանկալի է դասի մեթոդը միշտ հայտարարել `const`, եթե այն փոփոխության չի ենթարկում ընթացիկ օբյեկտը, այլապես այդ մեթոդը հնարավոր չի լինի կանչել հաստատուն օբյեկտների համար: Ստորև բերված է `Rational` դասի ճշգտրված սահմանումը՝

```
class Rational
{
public:
```

```
    /*այն բոլոր մեթոդները, որոնք չեն փոխում ընթացիկ օբյեկ-
     տը պետք է լինեն const մեթոդներ*/
    void Print() const;
    void SetNumerator(int p);
    int GetNumerator() const;
    void SetDenominator(int q);
    int GetDenominator() const;
private:
    int numerator;
    int denominator;
};
```

Դիտարկենք ևս մեկ օրինակ՝

```
class A
{
public:
    A():a(10),b(10){
        b=20;
        /*այս տողում սխալ կա, քանի որ const դաշտը
        չի կարող փոփոխվել*/
        a=20;
    }
    void f() const{
        /*այս տողում սխալ կա, քանի որ const
        մեթոդը չի կարող փոխել ընթացիկ օբյեկտը*/
        ++b;
        /*այս տողում սխալ կա, քանի որ const
        մեթոդը չի կարող կանչել ոչ const մեթոդ
        ընթացիկ օբյեկտի համար*/
        g();
    }
    void g(){
        ++b;
    }
```

```
private:
    const int a;
    int b;
};
```

Այժմ անդրադառնանք դասի ստատիկ անդամներին: Դասի անդամը համարվում է ստատիկ, եթե այդ անդամի նկարագրության մեջ դիմացից գրված է `static` ծառայողական բառը: Ստատիկ դաշտերը տարբերվում են ոչ ստատիկ դաշտերից նրանով, որ նրանք որևէ օբյեկտի մաս չեն կազմում, այլ ընդհանուր են դասի բոլոր օբյեկտների համար: Բացի այդ ստատիկ դաշտերը պետք է սկզբնարժեքավորել դասից դուրս՝

```
class A
{
    …
    static int a;
    …
};

//սկզբնարժեքավորում ենք ստատիկ դաշտը
int A::a=10;
```

Ոչ ստատիկ դաշտերի նման դասի ստատիկ դաշտը նույնպես կարելի է հայտարարել որպես `const` դաշտ, որն արգելում է փոփոխել դաշտի արժեքը: Ստատիկ մեթոդները տարբերվում են ոչ ստատիկ մեթոդներից նրանով, որ ստատիկ մեթոդները կանչվում են առանց ընթացիկ օբյեկտի: Որպես հետևանք՝ ստատիկ մեթոդներում `this` ցուցիչը չի կարելի օգտագործել: Չնայած ստատիկ անդամները ընդհանուր են դասի բոլոր օբյեկտների համար, սակայն նրանց հնարավոր է դիմել նաև ոչ ստատիկ անդամների նման՝ օգտագործելով `.` և `->` գործողությունները: Բացի

այդ դասի ստատիկ անդամներին՝ կարելի է դիմել նաև՝ օգտագործելով միայն դասի անունը՝

դաս: : *անդամ*

անդամ-ը *դաս*-ի ստատիկ անդամն է, որին դիմում ենք:

Գործողությունների վերաբեռնում

C++ լեզուն հնարավորություն է տալիս օգտագործել լեզվի գործողությունները (`+`, `*`, `%`, `==`, `>>`, `!`, `[]` և այլն) դասերի օբյեկտների նկատմամբ: Դա անելու համար պետք է տվյալ դասի համար վերաբեռնել համապատասխան գործողությունը: Գործողությունների վերաբեռնման հնարավորությունն ունի հետևյալ սահմանափակումները՝

- Հնարավոր չէ վերաբեռնել լեզվում գոյություն չունեցող գործողություն (օրինակ՝ `#`):
- Հնարավոր չէ փոփոխել վերաբեռնվող գործողության տեղայնությունը, առաջնայնությունը և ասոցիատիվությունը:
- Հետևյալ գործողությունները չի կարելի վերաբեռնել՝ `::`, `.`, `.*` և `?:`
- Գործողությունը կարելի է վերաբեռնել միայն այն դեպքում, երբ օպերանդներից գոնե մեկը դասի օբյեկտ է: Այսինքն, գործողությունը չի կարելի վերասահմանել լեզվի ներդրված տիպերի համար:

Գործողությունը վերաբեռնելու համար պետք է իրականացնել համապատասխան ֆունկցիա, որը կարող է լինել դասի մեթոդ կամ գլոբալ ֆունկցիա: Բացառություն են միայն `=`, `()`, `[]` և `->` գործողությունները, որոնք կարելի է վերաբեռնել միայն դասի մեթոդի միջոցով:

Ենթադրենք ունենք **θ** ունար գործողությունը, որը ցանկանում ենք վերաբեռնել այն դեպքում, երբ գործողության օպերան-

դը `T` դասի օբյեկտն է: Դա կարող ենք անել երկու տարբերակով (ոչ պարտադիր մասերը բերված են քառակուսի փակագծերում)՝

- իրականացնելով դասի հետևյալ մեթոդը՝

```
class T
{
    ...
    RType operator θ() [const];
    ...
};
```

Այս դեպքում մեթոդը ոչ մի պարամետր չի ստանում, և **θ** գործողության օպերանդը լինում է ընթացիկ օբյեկտը: `RType`-ը գործողության արդյունքում վերադարձվող արժեքի տիպն է, որը կարող է լինել ցանկացած տիպ (նաև `void`):

- իրականացնել հետևյալ գլոբալ ֆունկցիան՝

```
RType operator θ([const] T[&] op);
```

Այս դեպքում **θ** գործողության օպերանդը փոխանցվում է ֆունկցիայի պարամետրի միջոցով:

Երբ ծրագրում հանդիպում է **θ**`t` արտահայտությունը (որտեղ `t`-ն `T` դասի օբյեկտ է), ապա այդ արտահայտության փոխարեն տեղադրվում է համապատասխան ֆունկցիայի կանչը. առաջին դեպքում՝ `t.operator`**θ**`()`, իսկ երկրորդ դեպքում՝ `operator`**θ**`(t)`:

Ենթադրենք ունենք **θ** բինար գործողությունը, որը ցանկանում ենք վերաբեռնել այն դեպքում, երբ գործողության օպերանդներից մեկը `T` դասի օբյեկտն է: Բացի վերևում նշված բացառություններից, դա կարող ենք անել երկու տարբերակով (ոչ պարտադիր մասերը բերված են քառակուսի փակագծերում)՝

- իրականացնելով դասի հետևյալ մեթոդը՝

```
class T
{
    ...
    RType operator𝛉([const] Type[&] op2)
                                        [const];
    ...
};
```

Այս դեպքում մեթոդը պարամետրի միջոցով ստանում է գործողության երկրորդ օպերանդը (որը `Type` տիպի է), իսկ ընթացիկ օբյեկտը լինում է գործողության առաջին օպերանդը: `RType`-ը գործողության արդյունքում վերադարձվող արժեքի տիպն է, որը կարող է լինել ցանկացած տիպ (նաև `void`): Վերաբեռնման այս տարբերակի դեպքում `T` դասի օբյեկտը պետք է լինի գործողության առաջին օպերանդը:

- իրականացնելով հետևյալ գլոբալ ֆունկցիան՝

```
RType operator𝛉([const] Type1[&] op1,
                [const] Type2[&] op2);
```

Այս դեպքում **𝛉** գործողության օպերանդները (համապատասխանաբար՝ `Type1` և `Type2` տիպերի) փոխանցվում են ֆունկցիայի պարամետրերի միջոցով: Ընդ որում՝ նրանցից մեկը պետք է լինի `T` տիպի: Վերաբեռնման այս տարբերակի դեպքում `T` դասի օբյեկտը կարող է լինել գործողության և՛ առաջին, և՛ երկրորդ օպերանդը:

Երբ ծրագրում հանդիպում է `t1`**𝛉**`t2` արտահայտությունը (որտեղ `t1`-ը կամ `t2`-ը `T` դասի օբյեկտ է), ապա այդ արտահայտության փոխարեն տեղադրվում է համապատասխան ֆունկցի-

այի կանչը. առաջին դեպքում՝ `t1.operator`**θ**`(t2)`, ընդ որում՝ `t1`-ը պետք է լինի `T` դասի օբյեկտ, իսկ երկրորդ դեպքում՝ `operator`**θ**`(t1, t2)`:

Այժմ դիտարկենք գործողությունների վերաբեռնման մի քանի առանձնահատուկ դեպքեր՝

- Եթե ցանկանում ենք վերաբեռնել ունար գործողության ետադիր տարբերակը (C++ լեզվում կա այդպիսի երկու գործողություն՝ `++` և `--`), ապա համապատասխան ֆունկցիան պետք է ստանա լրացուցիչ ձևական `int` տիպի պարամետր՝

```
class T
{
    …
    RType operator θ(int) [const];
    …
};
```

կամ

```
RType operator θ([const] T[&] op, int);
```

Երբ ծրագրում հանդիպում է `t`**θ** արտահայտությունը (որտեղ `t`-ն `T` դասի օբյեկտ է), ապա այդ արտահայտության փոխարեն տեղադրվում է համապատասխան ֆունկցիայի կանչը. առաջին դեպքում՝ `t.operator`**θ**`(0)`, իսկ երկրորդ դեպքում՝ `operator`**θ**`(t, 0)`:

- Վերագրման գործողությունը հնարավոր է վերաբեռնել մյուս բինար գործողությունների նման, սակայն երբ այն վերաբեռնված չէ, ապա կոմպիլյատորը վերաբեռնում է այն (եթե հնարավոր է)՝ գեներացնելով հետևյալ մեթոդը՝

```
class T
{
    …
    T& operator=(const T& op2);
    …
};
```

Կոմպիլյատորի կողմից գեներացված մեթոդը ընթացիկ օբյեկտի բոլոր դաշտերին վերագրում է երկրորդ օպերանդի համապատասխան դաշտերը: Եթե որևէ դաշտի համար հնարավոր չէ դա անել (օրինակ՝ դաշտերից որևէ մեկը հաստատուն է), ապա կոմպիլյատորը չի վերաբեռնում վերագրման գործողությունը:

- () գործողության առանձնահատկությունն է, որ այն կարող է ունենալ կամայական քանակի օպերանդներ (զրոյից մեծ)՝ $t_1(t_2, \dots, t_n), n \geq 1$: Այն վերաբեռնելու համար պետք է համապատասխան մեթոդը կամ գլոբալ ֆունկցիան ունենան համապատասխան քանակի պարամետրեր՝

```
class T
{
    …
    RType operator()([const] Type2[&] op2,…,
                     [const] Typen[&] opn)
                                    [const];
    …
};
```

կամ

```
RType operator()([const] Type1[&] op1,…,
                 [const] Typen[&] opn);
```

- Եթե անհրաժեշտ է իրականացնել տիպի ձևափոխություն `T` տիպից դեպի `T1` տիպ, ապա դա կարելի է անել վերաբեռնելով տիպի ձևափոխության գործողությունը `T` դասում: Տիպի ձևափոխության գործողությունը կարելի է վերաբեռնել որպես դասի մեթոդ: Ընդ որում՝ այդ մեթոդի համար վերադարձվող արժեքի տիպը ֆիքսված է և չի նշվում՝

```
class T
{
    …
    [explicit] operator T1() [const];
    …
};
```

Եթե չենք ցանկանում, որ վերաբեռնված գործողությունն օգտագործվի `T` տիպից դեպի `T1` տիպ անբացահայտ ձևափոխության համար, ապա մեթոդի նկարագրության դիմացից ավելացնում ենք `explicit` ծառայողական բառը:

Որպես օրինակ վերաբեռնենք մի քանի գործողությունների `Rational` դասի համար՝

```
class Rational
{
    …
    Rational& operator+=(const Rational& op2){
        numerator*=op2.denominator;
        numerator+=op2.numerator*denominator;
        denominator*=op2.denominator;
        return *this;
    }
    Rational operator+(Rational op2) const{
        op2+=*this;
        return op2;
```

```
    }
    Rational& operator++(){
        *this+=Rational(1,1);
        return *this;
    }
    Rational operator++(int){
        Rational result=*this;
        ++*this;
        /*վերադարձնում ենք օբյեկտի արժեքը մինչև մեկ
        գումարելը*/
        return result;
    }
    bool operator==(const Rational& op2) const{
        return numerator*op2.denominator==
               denominator*op2.numerator;
    }
    bool operator>(const Rational& op2) const{
        return numerator*op2.denominator>
               denominator*op2.numerator;
    }
    operator double() const{
        return numerator/(double)denominator;
    }
    ...
};
```

Այս և հաջորդ բաժնի ոչ թեստային առաջադրանքներում պահանջվում է նախագծել և իրականացնել տրված դասը/դասերը, ինչպես նաև պահանջվում է լուծել մի քանի փոքր խնդիրներ՝ օգտագործելով այդ դասը/դասերը: Ենթադրվում է, որ

- Իրականացված դասերի բոլորը դաշտերը (անդամ փոփոխականները) պետք է լինեն փակ:
- «Մեթոդներ և գլոբալ ֆունկցիաներ» մասում նշված ֆունկցիաները պետք է իրականացվեն կա՛մ որպես դասի անդամ

ֆունկցիա, կա՛մ որպես գլոբալ ֆունկցիա (ըստ անհրաժեշտության): Ընդ որում՝ գլոբալ ֆունկցիաները կարող են անհրաժեշտության դեպքում հայտարարվել որպես դասի բարեկամ ֆունկցիաներ:

- Իրականացված դասերի բոլոր մեթոդներում (ներառյալ կոնստրուկտորները) պետք է կատարվեն բոլոր այն անհրաժեշտ ստուգումները/գործողությունները, որոնք կապահովեն դասի դաշտերի ճիշտ և ոչ հակասական արժեքներ ունենալը ցանկացած պահի:

Այժմ, օգտագործելով դասերը, լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Իրականացնել `Polynom` դասը, որի օբյեկտները իրական գործակիցներով բազմանդամներ են:

Դաշտեր.

- `double` տիպի զանգված՝ բազմանդամի գործակիցները պահելու համար,
- բազմանդամի աստիճանը,1
- `Polynom` տիպի ստեղծված օբյեկտների քանակը:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- առանց պարամետրի կոնստրուկտոր, որը ստեղծում է `0` գործակիցներով ու աստիճանով բազմանդամ,
- կոնստրուկտոր՝ `int` տիպի պարամետրով, որը ստեղծում է տրված աստիճանի բազմանդամ, որի բոլոր գործակիցները հավասար են `1`-ի,
- կոնստրուկտոր, որը ստեղծում է բազմանդամ՝ ստանալով դրա գործակիցները տրված `double` տիպի զանգվածից,
- պատճենման կոնստրուկտոր,
- դեստրուկտոր,

- ծրագրի աշխատանքի ընթացքում ստեղծված `Polynom` տիպի օբյեկտների քանակը վերադարձնող ֆունկցիա,
- `operator=` (բազմանդամների վերագրման գործողություն),
- `operator[]` (հնարավորություն է տալիս ըստ կարգահամարի օգտագործել բազմանդամի գործակիցը որպես և՛ աջակողմյան, և՛ ձախակողմյան օպերանդ՝ ստուգելով կարգահամարի թույլատրելի լինելը),
- `operator()` (հաշվում է բազմանդամի արժեքը տրված կետում),
- `operator<<` (բազմանդամն արտածելու ֆունկցիա),
- `operator+`, `operator*` (բազմանդամների գումարը և արտադրյալը հաշվող ֆունկցիաներ),
- `operator+=`, `operator*=` (բազմանդամների գումարը և արտադրյալը ձախ օպերանդում ստացող ֆունկցիաներ),
- բազմանդամի տրված կարգի ածանցյալը հաշվող ֆունկցիա,
- բազմանդամի աստիճանը իջեցնող փակ ֆունկցիա:

Գրել թեստ.

Նկարագրել `Polynom` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք:

Լուծում՝

```
#include <iostream>
#include <cassert>

using std::ostream;
using std::cout;
using std::endl;

class Polynom
{
```

```
    /*բազմանդամն արտածող ֆունկցիայի friend
    հայտարարություն*/
    friend ostream& operator<<(ostream& out,
                               const Polynom& p);
public:
    //առանց պարամետրի կոնստրուկտոր
    Polynom();
    /*տրված աստիճանի (1 գործակիցներով)
    բազմանդամի ստեղծում*/
    Polynom(int n);
    /*գործակիցների տրված զանգվածից բազմանդամի
    ստեղծում*/
    Polynom(double* coeff,int count);
    //պատճենման կոնստրուկտոր
    Polynom(const Polynom& other);
    ~Polynom(){ delete []coeff; } //դեստրուկտոր
    /*ստեղծված օբյեկտների քանակը վերադարձնող
    ֆունկցիա*/
    static int GetCreatedObjectsCount()
    { return objCount; }
    //բազմանդամների վերագրում
    Polynom& operator=(const Polynom& other);
    //բազմանդամների գումարում
    Polynom operator+(const Polynom& other)const;
    //բազմանդամների գումարում
    const Polynom& operator+=(const Polynom&
                                        other);
    //բազմանդամների բազմապատկում
    Polynom operator*(const Polynom& other)
                                        const;
    //բազմանդամների բազմապատկում
```

```
    const Polynom& operator*=(const Polynom&
                                          other);
    /*X-ի տրված աստիճանի գործակիցը վերադարձնող
    ֆունկցիա*/
    double operator[](int i) const
    { return coeff[i>=0?i:0]; }
    /*X-ի տրված աստիճանի գործակցի հղումը
    վերադարձնող ֆունկցիա*/
    double& operator[](int i);
    /*բազմանդամի արժեքը տրված կետում հաշվող
    ֆունկցիա*/
    double operator()(double x) const;
    /*բազմանդամի տրված կարգի ածանցյալը հաշվող
    ֆունկցիա*/
    Polynom Derivative(int order=1) const;
private:
    /*բազմանդամի աստիճանը անհրաժեշտության
    դեպքում իջեցնող ֆունկցիա (երբ սկզբի մեկից ավելի
    գործակիցներ զրոներ են)*/
    void correctDegree();
    //գործակիցների դինամիկ զանգվածի հասցեն
    double *coeff;
    int degree; //բազմանդամի աստիճանը
    //ստեղծված օբյեկտների քանակը
    static int objCount;
};

int Polynom::objCount=0;

Polynom::Polynom()
    :coeff(new double[1]),degree(0)
{
```

```
    /*լռությամբ կոնստրուկտորը ստեղծում է 0-ին
    հավասար բազմանդամ*/
    coeff[0]=0;
    ++objCount;
}

Polynom::Polynom(int n)
{
    /*n<0 դեպքերում ստեղծում ենք 0 աստիճանի
    բազմանդամ*/
    degree=n<0?0:n;
    coeff=new double[degree+1];
    //իմաստ ունի միայն Debug ռեժիմիում
    assert(coeff);
    for(int i=0;i<=degree;++i)
    {
        /*ստեղծում ենք 1 գործակիցներով տրված
        աստիճանի բազմանդամ*/
        coeff[i]=1;
    }
    ++objCount;
}

Polynom::Polynom(double* coeff,int count)
{
    if(coeff==0)
    {
        /*զանգվածի հասցեն 0 լինելու դեպքում
        ստեղծում ենք 0-ին հավասար բազմանդամ */
        degree=0;
        this->coeff=new double[1];
        this->coeff[0]=0;
        return;
```

```
    }
    if(count<=0) {count=1;}
    degree=count-1;
    this->coeff=new double[degree+1];
    //իմաստ ունի միայն Debug ռեժիմիում
    assert(coeff);
    for(int i=0;i<=degree;++i)
    {
        //գործակիցները վերցնում ենք տրված զանգվածից
        this->coeff[i]=coeff[i];
    }
    ++objCount;
    //հեռացնում ենք սկզբի ավելորդ 0 գործակիցները
    correctDegree();
}

Polynom::Polynom(const Polynom& other)
    :degree(other.degree)
{
    coeff=new double[degree+1];
    //իմաստ ունի միայն Debug ռեժիմիում
    assert(coeff);
    for(int i=0;i<=degree;++i)
    {
        /*գործակիցները վերցնում ենք տրված
        բազմանդամից*/
        coeff[i]=other.coeff[i];
    }
    ++objCount;
}

Polynom& Polynom::operator=(const Polynom &other)
{
```

```
    //բացառում ենք ինքն իրեն վերագրելու դեպքը
    if(&other!=this)
    {
        if(degree!=other.degree)
        {
            delete []coeff;
            degree=other.degree;
            coeff=new double[degree+1];
            //իմաստ ունի միայն Debug ռեժիմիում
            assert(coeff);
        }
        for(int i=0;i<=degree;++i)
        {
            /*գործակիցները վերցնում ենք տրված
            բազմանդամից*/
            coeff[i]=other.coeff[i];
        }
    }
    return *this;
}

const Polynom& Polynom::operator+=(const Polynom
&other)
{
    if(degree<other.degree)
    {
        /*պետք է մեծացնել գործակիցների զանգվածի
        չափը*/
        double* coeffNew=
                    new double[other.degree+1];
        //իմաստ ունի միայն Debug ռեժիմիում
        assert(coeffNew);
        int i=0;
```

```
        for(;i<=degree;++i)
        {
            coeffNew[i]=coeff[i];
        }
        for(;i<=other.degree;++i)
        {
            coeffNew[i]=0;
        }
        delete []coeff;
        coeff=coeffNew;
        degree=other.degree;
    }
    for(int i=0;i<=other.degree;++i)
    {
        /*երկրորդ բազմանդամի գործակիցը գումարում
        ենք առաջին բազմանդամի համապատասխան
        գործակցին*/
        coeff[i]+=other.coeff[i];
    }
    //հեռացնում ենք սկզբի ավելորդ 0 գործակիցները
    correctDegree();
    return *this;
}

Polynom Polynom::operator+(const Polynom &other)
const
{
    Polynom result(*this);
    //օգտագործում ենք += գործողության իրականացումը
    result+=other;
    return result;
}
```

```
const Polynom& Polynom::operator*=(const Polynom
&other)
{
    //արդյունքի գործակիցների զանգվածը
    double* coeffNew=
            new double[other.degree+degree+1];
    //իմաստ ունի միայն Debug ռեժիմիում
    assert(coeffNew);
    for(int i=0;i<=other.degree+degree;++i)
    {
        coeffNew[i]=0;
    }
    for(int i=0;i<=degree;++i)
    {
        for(int j=0;j<=other.degree;++j)
        {
            coeffNew[i+j]+=coeff[i]*
                                other.coeff[j];
        }
    }
    delete []coeff;
    coeff=coeffNew;
    degree+=other.degree;
    //հեռացնում ենք սկզբի ավելորդ 0 գործակիցները
    correctDegree();
    return *this;
}

Polynom Polynom::operator*(const Polynom &other)
const
{
    Polynom result(*this);
    //օգտագործում ենք *= գործողության իրականացումը
    result*=other;
```

```
    return result;
}

double& Polynom::operator[](int i)
{
    //պե՞տք է բարձրացնել բազմանդամի աստիճանը
    if(i>degree)
    {
        //գործակիցների մեծացված զանգվածը
        double* coeffNew=new double[i+1];
        //իմաստ ունի միայն Debug ռեժիմիում
        assert(coeffNew);
        for(int j=0;j<=degree;++j)
        {
            coeffNew[j]=coeff[j];
        }
        for(int j=degree+1;j<=i;++j)
        {
            coeffNew[j]=0;
        }
        delete[] coeff;
        coeff=coeffNew;
        degree=i;
    }
    return coeff[i>=0?i:0];
}

double Polynom::operator()(double x) const
{
    double result=coeff[degree];
    for(int i=degree-1;i>=0;--i)
    {
        result=result*x+coeff[i];
    }
```

```
    return result;
}

Polynom Polynom::Derivative(int order) const
{
    if(order<=0){order=1;}
    if(degree<order)
    {
        //ածանցյալը հավասար է 0-ի
        return Polynom();
    }
    else
    {
        //պարունակելու է ածանցյալի գործակիցները
        double* coeffNew=
                    new double[degree+1-order];
        //իմաստ ունի միայն Debug ռեժիմում
        assert(coeffNew);
        for(int i=0;i<=degree-order;++i)
        {
            coeffNew[i]=coeff[i+order];
            for(int j=i+order;j>i;--j)
            {
                coeffNew[i]*=j;
            }
        }
        return Polynom(coeffNew,degree+1-order);
    }
}

void Polynom::correctDegree()
{
    // դիմացի 0 գործակիցների քանակը
    int zCount=0;
```

```
    for(;zCount<=degree &&
            coeff[degree-zCount]==0;++zCount);
    if(zCount==degree+1)
    {
        //բոլոր գործակիցները 0 են
        if(degree!=0)
        {
            delete[] coeff;
            degree=0;
            coeff=new double[1];
            //իմաստ ունի միայն Debug ռեժիմիում
            assert(coeff);
            coeff[0]=0;
        }
    }
    else
    {
        /*պարունակելու է գործակիցները առանց դիմացի
        0-ների*/
        double* coeffNew=
                    new double[degree+1-zCount];
        //իմաստ ունի միայն Debug ռեժիմիում
        assert(coeffNew);
        for(int i=0;i<=degree-zCount;++i)
        {
            coeffNew[i]=coeff[i];
        }
        degree-=zCount;
        delete[] coeff;
        coeff=coeffNew;
    }
}
```

```
//բազմանդամն արտածող գլոբալ ֆունկցիա
ostream &operator<<(ostream& out,
                    const Polynom& p)
{
    //որևէ գումարելի տպվե՞լ է
    bool anyMemberPrinted=false;
    for(int i=p.degree;i>=0;--i)
    {
        if(p.coeff[i]==0)
        {
            /*0 գործակցով անդամները կարիք չկա
            արտածելու*/
            continue;
        }
        double currentCoef=p.coeff[i];
        if(anyMemberPrinted || currentCoef<0)
        {
            out<<(currentCoef>0?'+':'-');
            if(currentCoef<0)
            {
                currentCoef=-currentCoef;
            }
        }
        anyMemberPrinted=true;
        if(currentCoef!=1)
        {
            out<<currentCoef<<(i>0?"*X":"");
        }
        else
        {
            out<<(i>0?"X":"1");
        }
```

```
        if(i>1)
        {
            out<<'^'<<i;
        }
    }
    if(!anyMemberPrinted)
    {
        out<<0;
    }
    return out;
}

int main()
{
    cout<<"Objects Count="
        <<Polynom::GetCreatedObjectsCount()
        <<endl;
    double coeff[4]={1.1,2.0,-1,-1};
    Polynom p1(4),p2(coeff,4),p3(p2),p4,
            p5(coeff,3);
    cout<<"p1="<<p1<<endl;
    cout<<"p2="<<p2<<endl;
    cout<<"p3="<<p3<<endl;
    cout<<"p4="<<p4<<endl;
    cout<<"p5="<<p5<<endl;
    cout<<"Objects Count="
        <<Polynom::GetCreatedObjectsCount()
        <<endl;
    Polynom p6=p2.Derivative();
    Polynom p7=p2.Derivative(2);
    cout<<"\nAfter p6=p2.Derivative()"
        <<" and p7=p2.Derivative(2)\n";
    cout<<"p6="<<p6<<endl<<"p7="<<p7<<endl;
    cout<<"Objects Count="
```

```
        <<Polynom::GetCreatedObjectsCount()
        <<endl;
    p5=p4=p3;
    cout<<"\nAfter p5=p4=p3\n";
    cout<<"p3="<<p3<<endl<<"p4="<<p4<<endl
        <<"p5="<<p5<<endl;
    cout<<"Objects Count="
        <<Polynom::GetCreatedObjectsCount()
        <<endl;
    p4=p1+p2;
    cout<<"\nAfter p4=p1+p2\n";
    cout<<"p4="<<p4<<endl<<"p4(2)="<<p4(2)<<endl;
    cout<<"Objects Count="
         <<Polynom::GetCreatedObjectsCount()
        <<endl;
    p7[2]=2.5;
    p4+=p7;
    cout<<"\nAfter p7[2]=2.5 and p4+=p7\n";
    cout<<"p4="<<p4<<endl<<"p7="<<p7<<endl
         <<"p4(2)="<<p4(2)<<endl;
    cout<<"Objects Count="
        <<Polynom::GetCreatedObjectsCount()
        <<endl;
    p4=p1*p2;
    p7*=p1;
    cout<<"\nAfter p4=p1*p2 and p7*=p1\n";
    cout<<"p4="<<p4<<endl<<"p7="<<p7<<endl;
    cout<<"Objects Count="
        <<Polynom::GetCreatedObjectsCount()
        <<endl;
    return 0;
}
```

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն:

1. Ի՞նչ քերականական սխալներ կան ծրագրի հետևյալ հատվածում.

ա)
```
class T{
public:
    T(const T& other){ other.g(); }
    int g(){}
    void operator.(T& other){ other.g(); }
};
```

բ)
```
class A{
public:
    int* f() const{ return &c; }
    void ~A(){}
private:
    int c;
    int d=15;
};
```

գ)
```
class B{
public:
    void f() const { ++b; g(); }
    void g(){}
private:
    static int b=20;
};
```

2. Իրականացնել `NoteBook` դասը, որի օբյեկտները տետրեր են:
   Դաշտեր.
   - տետրի թերթերի քանակը (`int` տիպի),
   - տետրի գինը (`int` տիպի),
   - տետրի լրացված էջերի քանակը (`int` տիպի):

Մեթոդներ և գլոբալ ֆունկցիաներ.

- առանց պարամետրի կոնստրուկտոր, որը զրոյացնում է դասի դաշտերը,
- կոնստրուկտոր՝ երկու `int` տիպի պարամետրով, որը ստեղծում է տրված թերթերի քանակով և գնով տետր,
- տետրի թերթերի քանակը, գինը, լրացված էջերի քանակը ստանալու և փոփոխելու ֆունկցիաներ,
- տետրը ներածելու և արտածելու ֆունկցիաներ,
- տետրի մաքուր էջերի քանակը հաշվող ֆունկցիա,
- տետրերի հավասարությունը ստուգող ֆունկցիա:

Գրել թեստ.

- նկարագրել `NoteBook` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `NoteBook` տիպի օբյեկտների զանգված և արտածել այն տետրերը, որոնց մաքուր էջերի քանակը մեծ է տրված թվից,
- ներածել `NoteBook` տիպի օբյեկտների զանգված և արտածել այն ամենաթանկ տետրը, որի էջերի կեսից ավելին մաքուր են:

3. Իրականացնել `Point` դասը, որի օբյեկտները երկչափ տարածության կետեր են:
   Դաշտեր.
   - կետի կոորդինատները (`double` տիպի):

   Մեթոդներ և գլոբալ ֆունկցիաներ.
   - առանց պարամետրի կոնստրուկտոր, որը ստեղծում է `(0,0)` կոորդինատներով կետ,
   - կոնստրուկտոր՝ երկու `double` տիպի պարամետրով, որը ստեղծում է տրված կոորդինատներով կետ,
   - կետի ներածման և արտածման ֆունկցիաներ,

- երկու կետերի հեռավորությունը հաշվող ֆունկցիա,
- կետի հեռավորությունը կոորդինատների սկզբնակետից հաշվող ֆունկցիա,
- `operator==`, `operator!=` (կետերի համեմատման ֆունկցիաներ),
- կետը `x`-երի առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,
- կետը `y`-ների առանցքին զուգահեռ տեղաշարժելու ֆունկցիա:

Գրել թեստ.

- նկարագրել `Point` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Point` տիպի օբյեկտների զանգված և արտածել այն կետերը, որոնց հեռավորությունը կոորդինատների սկզբնակետից մեծ է տրված իրական թվից,
- ներածել `Point` տիպի օբյեկտների զանգված և արտածել երկու իրարից ամենահեռու (ամենամոտիկ) կետերը:

4. Իրականացնել `Rectangle` դասը, որի օբյեկտներն ուղղանկյուններ են:

   Դաշտեր.

   - ուղղանկյան կողմերի երկարությունները (`double` տիպի):

   Մեթոդներ և գլոբալ ֆունկցիաներ.

   - կոնստրուկտոր՝ երկու `double` տիպի պարամետրով, որը ստեղծում է տրված կողմերով ուղղանկյուն (պարամետրերին տալ լռությամբ `1` արժեք),
   - ուղղանկյան պարագիծը և մակերեսը հաշվող ֆունկցիաներ,
   - ուղղանկյունն արտածելու և ներածելու ֆունկցիաներ,

- `operator==`, `operator!=` (ուղղանկյունների համեմատման ֆունկցիաներ):

Գրել թեստ.

- նկարագրել `Rectangle` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Rectangle` տիպի օբյեկտների զանգված և արտածել այն ուղղանկյունները, որոնց մակերեսը մեծ է տրված իրական թվից,
- ներածել `Rectangle` տիպի օբյեկտների զանգված և արտածել ուղղանկյունները՝ իրենց պարագծերի նվազման կարգով:

5. Իրականացնել `Rectangle` դասը, որի օբյեկտները երկչափ տարածության կոորդինատային առանցքներին զուգահեռ կողմերով ուղղանկյուններ են:
   Դաշտեր.
   - ուղղանկյան վերին ձախ գագաթի կոորդինատները (`double` տիպի),
   - ուղղանկյան կողմերի երկարությունները (`double` տիպի):

   Մեթոդներ և գլոբալ ֆունկցիաներ.
   - կոնստրուկտոր՝ չորս `double` տիպի պարամետրով, որը ստեղծում է վերին ձախ գագաթի տրված կոորդինատներով և տրված կողմերով ուղղանկյուն (պարամետրերին տալ լռությամբ `0` արժեք),
   - ուղղանկյունն արտածելու և ներածելու ֆունկցիաներ,
   - երկու ուղղանկյունների հատումը վերադարձնող ֆունկցիա,
   - ուղղանկյունը `x`-երի առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,

- ուղղանկյունը `y`-ների առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,
- վերին ձախ գագաթի շուրջը `90`, `180` և `270` աստիճանով ուղղանկյան պտույտն ապահովող ֆունկցիաներ,
- `operator<` (երկու ուղղանկյունների համեմատման ֆունկցիա՝ ըստ իրենց մակերեսների):

Գրել թեստ.

- նկարագրել `Rectangle` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Rectangle` տիպի օբյեկտների զանգված և արտածել զանգվածի բոլոր տարրերի հատման արդյունքում ստացված ուղղանկյունը կամ `NO`, եթե բոլոր ուղղանկյունների հատումը դատարկ է,
- ներածել `Rectangle` տիպի օբյեկտների զանգված և արտածել զանգվածի այն երկու տարրերը, որոնց հատման մակերեսն առավելագույնն է:

6. Իրականացնել `Circle` դասը, որի օբյեկտները երկչափ տարածության շրջաններ են:
   Դաշտեր.
   - կենտրոնի կոորդինատները (`double` տիպի),
   - շրջանի շառավիղը (`double` տիպի):

   Մեթոդներ և գլոբալ ֆունկցիաներ.
   - կոնստրուկտոր՝ երեք `double` տիպի պարամետրով, որը ստեղծում է տրված շառավղով և կենտրոնի տրված կոորդինատներով շրջան (պարամետրերին տալ լռությամբ արժեքներ),
   - շրջանն արտածելու և ներածելու ֆունկցիաներ,
   - շրջանի մակերեսը և պարագիծը հաշվող ֆունկցիաներ,

- երկու շրջանների կենտրոնների հեռավորությունը հաշվող ֆունկցիա,
- `operator==`, `operator!=` (շրջանների համեմատման ֆունկցիաներ),
- `operator<`, `operator>` (շրջանների համեմատման ֆունկցիաներ՝ ըստ իրենց մակերեսների),
- շրջանը `x`-երի առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,
- շրջանը `y`-ների առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,
- շրջանի մակերեսը `k` անգամ մեծացնելու ֆունկցիա, որտեղ `k`-ն դրական իրական թիվ է:

Գրել թեստ.

- նկարագրել `Circle` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Circle` տիպի օբյեկտների զանգված և արտածել այն շրջանները, որոնց մակերեսը մեծ է տրված իրական թվից,
- ներածել `Circle` տիպի օբյեկտների զանգված և արտածել այն շրջանը, որը շոշափում է առավելագույն թվով շրջանների,
- ներածել `Circle` տիպի օբյեկտների զանգված և արտածել այն առավելագույն թվով շրջանները, որոնք ներդրված են մեկը մյուսի մեջ:

7. Իրականացնել `LineSegment` դասը, որի օբյեկտները երկչափ տարածության հատվածներ են:
   Դաշտեր.
   - հատվածի ծայրակետերի կոորդինատները (`double` տիպի):

Մեթոդներ և գլոբալ ֆունկցիաներ.

- կոնստրուկտոր՝ չորս `double` տիպի պարամետրով, որը ստեղծում է տրված ծայրակետերով հատված (պարամետրերին տալ լռությամբ արժեքներ),
- հատվածը ներածելու և արտածելու ֆունկցիաներ,
- հատվածի երկարությունը հաշվող ֆունկցիա,
- `operator==`, `operator!=` (հատվածների համեմատման ֆունկցիաներ),
- `operator<`, `operator>` (հատվածների համեմատման ֆունկցիաներ՝ ըստ իրենց երկարությունների),
- հատվածը `x`-երի առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,
- հատվածը `y`-ների առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,
- հատվածի՝ առանցքների նկատմամբ կազմած անկյունը հաշվող ֆունկցիաներ,
- հատվածը իր կենտրոնի շուրջը տրված աստիճանով պտտելու ֆունկցիա:

Գրել թեստ.

- նկարագրել `LineSegment` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `LineSegment` տիպի օբյեկտների զանգված և արտածել այն հատվածները, որոնց երկարությունը մեծ է տրված իրական թվից,
- ներածել `LineSegment` տիպի օբյեկտների զանգված և արտածել այդ հատվածները՝ `x`-երի առանցքի հետ կազմած անկյան աճման կարգով:

8. Իրականացնել `Line` դասը, որի օբյեկտները երկչափ տարածության ուղիղներ են:

Դաշտեր.

- ուղղի `y=kx+b` ներկայացման `k` և `b` գործակիցները (`double` տիպի):

Մեթոդներ և գլոբալ ֆունկցիաներ.

- կոնստրուկտոր՝ երկու `double` տիպի պարամետրով, որը ստեղծում է տրված `k` և `b` գործակիցներով ուղիղ (պարամետրերին տալ լռությամբ արժեքներ),
- ուղիղը ներածելու և արտածելու ֆունկցիաներ,
- `operator==, operator!=` (ուղիղների համեմատման ֆունկցիաներ),
- ուղղի և առանցքների հատման կոորդինատները հաշվող ֆունկցիաներ,
- երկու ուղիղների զուգահեռությունը ստուգող ֆունկցիա,
- երկու ուղիղների հատման կետի կոորդինատները հաշվող ֆունկցիա (պետք է նաև մշակել հատման միակ կետ գոյություն չունենալու դեպքը),
- ուղիղը `x`-երի առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,
- ուղիղը `y`-ների առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,
- ուղղի՝ առանցքների նկատմամբ կազմած անկյունը հաշվող ֆունկցիաներ:

Գրել թեստ.

- նկարագրել `Line` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Line` տիպի օբյեկտների զանգված և արտածել այն ուղիղները, որոնք զուգահեռ են տրված ուղղին,

- ներածել `Line` տիպի օբյեկտների զանգված և արտածել այն ուղիղը, որը հատվում է առավելագույն թվով ուղիղների հետ,
- ներածել `Line` տիպի օբյեկտների զանգված և արտածել այն կետերը, որոնք ստացվում են այդ ուղիղների՝ միմյանց հետ հատումների արդյունքում:

9. Իրականացնել `Triangle` դասը, որի օբյեկտները եռանկյուններ են:

   Դաշտեր.
   - եռանկյան կողմերի երկարությունները (`double` տիպի):

   Մեթոդներ և գլոբալ ֆունկցիաներ.
   - առանց պարամետրի կոնստրուկտոր, որը ստեղծում է `1` կողմերով հավասարակողմ եռանկյուն,
   - կոնստրուկտոր՝ երեք `double` տիպի պարամետրով, որը ստեղծում է տրված կողմերով եռանկյուն,
   - եռանկյունը ներածելու և արտածելու ֆունկցիաներ,
   - եռանկյան պարագիծը և մակերեսը հաշվող ֆունկցիաներ,
   - `operator==`, `operator!=` (եռանկյունների համեմատման ֆունկցիաներ),
   - եռանկյունների նմանությունը ստուգող ֆունկցիա,
   - եռանկյան անկյունները հաշվող ֆունկցիա,
   - ֆունկցիա, որը ստուգում է եռանկյունը ուղղանկյուն եռանկյուն է, թե՞ ոչ:

   Գրել թեստ.
   - նկարագրել `Triangle` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,

- ներածել `Triangle` տիպի օբյեկտների զանգված և արտածել այն ուղղանկյուն եռանկյունները, որոնց պարագիծը մեծ է տրված թվից,
- ներածել `Triangle` տիպի օբյեկտների զանգված և արտածել առավելագույն անկյուն ունեցող եռանկյունը:

10. Իրականացնել `Triangle` դասը, որի օբյեկտները երկչափ տարածություն եռանկյուններ են:
    Դաշտեր.
    - եռանկյան երեք գագաթների կոորդինատները (`double` տիպի):

    Մեթոդներ և գլոբալ ֆունկցիաներ.
    - կոնստրուկտոր՝ վեց `double` տիպի պարամետրով, որը ստեղծում է տրված գագաթներով եռանկյուն (պարամետրերին տալ լռությամբ արժեքներ),
    - եռանկյունը ներածելու և արտածելու ֆունկցիաներ,
    - եռանկյան պարագիծը և մակերեսը հաշվող ֆունկցիաներ,
    - `operator<`, `operator>` (եռանկյունների համեմատման ֆունկցիաներ՝ ըստ իրենց մակերեսների),
    - եռանկյունների նմանությունը ստուգող ֆունկցիա,
    - եռանկյունը `x`-երի առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,
    - եռանկյունը `y`-ների առանցքին զուգահեռ տեղաշարժելու ֆունկցիա,
    - եռանկյան հավասարակողմ լինելը ստուգող ֆունկցիա,
    - եռանկյան հավասարասրուն լինելը ստուգող ֆունկցիա:

    Գրել թեստ.
    - նկարագրել `Triangle` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,

- ներածել `Triangle` տիպի օբյեկտների զանգված և արտածել այն հավասարասրուն եռանկյունը, որը նման է առավելագույն թվով եռանկյունների,
- ներածել `Triangle` տիպի օբյեկտների զանգված և արտածել դրանք մակերեսների չնվազման կարգով:

11. Տրված է հետևյալ սահմանումը.
    `typedef enum Currency {AMD, RUB, USD };`
    Իրականացնել `Deposit` դասը, որի օբյեկտները բանկային ավանդներ են:
    Դաշտեր.
    - ավանդի սկզբնական գումարը և տարեկան տոկոսադրույքը (`double` տիպի),
    - ավանդի արժույթը (`Currency` տիպի),
    - `double` տիպի `2` երկարությամբ զանգված, որտեղ պահվում են արտարժույթների փոխանակման կուրսերը դրամով (պետք է լինի ստատիկ դաշտ):

    Մեթոդներ և գլոբալ ֆունկցիաներ.
    - կոնստրուկտոր, որը ստեղծում է տրված գումարով, տոկոսադրույքով և արժույթով ավանդ (պարամետրերին տալ լռությամբ արժեքներ),
    - ավանդի սկզբնական գումարը, տոկոսադրույքը և արժույթը ստանալու ֆունկցիաներ,
    - ավանդը ներածելու և արտածելու ֆունկցիաներ,
    - ֆունկցիա, որը վերադարձնում է ավանդի գումարը `k` տարի հետո,
    - ֆունկցիա, որը վերադարձնում է տոկոսների գումարը `k` տարի հետո,
    - `operator==` (ավանդների համեմատման ֆունկցիա),

- `operator>` (ավանդների համեմատման ֆունկցիա՝ ըստ իրենց սկզբնական գումարի),
- ֆունկցիաներ՝ տրված արտարժույթի համար փոխանակման կուրսերը ստանալու և փոփոխելու համար:

Գրել թեստ.

- նկարագրել `Deposit` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Deposit` տիպի օբյեկտների զանգված և արտածել այն ավանդները, որոնց գումարը մեծ է տրված թվից:

12. Իրականացնել `Cylinder` դասը, որի օբյեկտները գլաններ են:

Դաշտեր.

- գլանի հիմքի շրջանի շառավիղը և բարձությունը (`double` տիպի):

Մեթոդներ և գլոբալ ֆունկցիաներ.

- առանց պարամետրի կոնստրուկտոր, որը ստեղծում է միավոր շառավղով և բարձրությամբ գլան,
- կոնստրուկտոր՝ երկու `double` տիպի պարամետրով, որը ստեղծում է տրված հիմքի շառավղով և բարձրությամբ գլան,
- գլանը ներածելու և արտածելու ֆունկցիաներ,
- գլանի մակերևույթի մակերեսը և ծավալը հաշվող ֆունկցիաներ,
- `operator==`, `operator!=` (գլանների համեմատման ֆունկցիաներ),
- `operator<`, `operator>` (գլանների համեմատման ֆունկցիաներ՝ ըստ իրենց ծավալների),
- գլանը `k` անգամ ձգելու/սեղմելու ֆունկցիա, որտեղ `k`-ն դրական իրական թիվ է:

Գրել թեստ.

- նկարագրել `Cylinder` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Cylinder` տիպի օբյեկտների զանգված և արտածել այն գլանները, որոնց ծավալը փոքր է տրված թվից,
- ներածել `Cylinder` տիպի օբյեկտների զանգված և արտածել երեք մեծագույն մակերևույթի մակերես ունեցող գլանները:

13. Իրականացնել `Complex` դասը, որի օբյեկտները կոմպլեքս թվեր են:

    Դաշտեր.

    - կոմպլեքս թվի իրական և կեղծ մասերը (`double` տիպի):

    Մեթոդներ և գլոբալ ֆունկցիաներ.

    - կոնստրուկտոր՝ երկու `double` տիպի պարամետրով, որը ստեղծում է տրված իրական և կեղծ մասերով կոմպլեքս թիվ (պարամետրերին տալ լռությամբ արժեքներ),
    - կոմպլեքս թվի իրական և կեղծ մասերը ստանալու և փոփոխելու ֆունկցիաներ,
    - կոմպլեքս թվի մոդուլը հաշվող ֆունկցիա,
    - `operator==`, `operator!=` (կոմպլեքս թվերի համեմատման ֆունկցիաներ),
    - `operator+`, `operator-`, `operator*`, `operator/` (կոմպլեքս թվերի գումարը, տարբերությունը, արտադրյալը և հարաբերությունը հաշվող ֆունկցիաներ),
    - `operator+=`, `operator-=`, `operator*=`, `operator/=` (կոմպլեքս թվերի գումարը, տարբերությունը, արտադրյալը և հարաբերությունը ձախ օպերանդում ստացող ֆունկցիաներ),

- `operator++, operator--` (իրական մասի ինկրեմենտի ու դեկրեմենտի պոստֆիքս ու պրեֆիքս տարբերակները),
- `operator>>, operator<<` (կոմպլեքս թիվը ներածելու և արտածելու ֆունկցիաներ):

Գրել թեստ.

- նկարագրել `Complex` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Complex` տիպի օբյեկտների զանգված և արտածել այն կոմպլեքս թվերը, որոնց կեղծ մասը փոքր է տրված թվից,
- ներածել `Complex` տիպի օբյեկտների զանգված և արտածել դրանց գումարն ու արտադրյալը,
- ներածել `Complex` տիպի օբյեկտների զանգված և արտածել դրանք աճման կարգով՝ ըստ իրենց մոդուլի:

14. Իրականացնել `Rational` դասը, որի օբյեկտները ռացիոնալ թվեր են:
Դաշտեր.
    - `m/n` ռացիոնալ թվի համարիչը և հայտարարը, որտեղ `m`-ը ամբողջ թիվ է, իսկ `n`-ը՝ բնական (պետք է կոնստրուկտորում և բոլոր գործողություններից հետո կոտորակը բերել կրճատված տեսքի):

    Մեթոդներ և գլոբալ ֆունկցիաներ.
    - կոնստրուկտոր՝ երկու `int` տիպի պարամետրով, որը ստեղծում է տրված համարիչով և հայտարարով ռացիոնալ թիվ (առաջին պարամետրի համար `0`-ն, իսկ երկրորդ պարամետրի համար `1`-ը սահմանել որպես լռությամբ արժեքներ),
    - ռացիոնալ թվի համարիչը և հայտարարը ստանալու և փոփոխելու ֆունկցիաներ,

- ռացիոնալ թվերի համեմատման ֆունկցիաներ՝ `operator==`, `operator!=`, `operator<`, `operator<=`, `operator>`, `operator>=`,
- `operator+`, `operator-`, `operator*`, `operator/` (ռացիոնալ թվերի գումարը, տարբերությունը, արտադրյալը և հարաբերությունը հաշվող ֆունկցիաներ),
- `operator+=`, `operator-=`, `operator*=`, `operator/=` (ռացիոնալ թվերի գումարը, տարբերությունը, արտադրյալը և հարաբերությունը ձախ օպերանդում ստացող ֆունկցիաներ),
- `operator++`, `operator--` (ինկրեմենտի ու դեկրեմենտի պոստֆիքս ու պրեֆիքս տարբերակները),
- `operator>>`, `operator<<` (ռացիոնալ թիվը ներածելու և արտածելու ֆունկցիաներ),
- `operator int`, `operator double` (դեպի `int` և դեպի `double` տիպի ձևափոխություն իրականացնող ֆունկցիաներ),
- կոտորակը կրճատող փակ ֆունկցիա:

Գրել թեստ.

- նկարագրել `Rational` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Rational` տիպի օբյեկտների զանգված և արտածել այն թվերը, որոնք մեծ են տրված թվից,
- ներածել `Rational` տիպի օբյեկտների զանգված և արտածել դրանց գումարն ու արտադրյալը,
- ներածել `Rational` տիպի օբյեկտների զանգված և արտածել դրանցից մեծագույնն ու փոքրագույնը:

15. Իրականացնել `Credit` դասը, որի օբյեկտները բանկային վարկեր են:

Դաշտեր.

- վարկի գումարը և տարեկան տոկոսադրույքը՝ `double` տիպի (համարել, որ տարին պարունակում է `365` օր),
- վարկի մարման ժամկետը ամիսներով,
- վարկի մարման եղանակը.

  `0` - հավասարաչափ, երբ մինչև վարկի մարման ժամկետի ավարտը յուրաքանչյուր ամիս վճարվում է միևնույն գումարը (այսինքն մարված մայր գումարի և տոկոսագումարի գումարը նույնն է բոլոր ամիսների համար),

  `1` - անհավասարաչափ, երբ վարկի մայր գումարը վճարվում է ամսական հավասար կտորներով, իսկ ամսական տոկոսագումարները նվազում են վարկի մայր գումարի մարմանը զուգընթաց:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- կոնստրուկտոր, որը ստեղծում է տրված գումարով, տոկոսադրույքով, ժամկետով և մարման եղանակով վարկ (պարամետրերին տալ լռությամբ արժեքներ),
- վարկի գումարը, տոկոսադրույքը, ժամկետը և մարման եղանակը ստանալու և փոփոխելու ֆունկցիաներ,
- վարկը ներածելու և արտածելու ֆունկցիաներ,
- `k` օր հետո վարկի մայր գումարի մնացորդը և վճարված ընդհանուր գումարը հաշվարկող ֆունկցիաներ,
- `operator==`, `operator!=` (վարկերի համեմատման ֆունկցիաներ),
- `operator<`, `operator>` (վարկերի համեմատման ֆունկցիաներ՝ ըստ իրենց գումարի):

Գրել թեստ.

- նկարագրել `Credit` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,

- ներածել `Credit` տիպի օբյեկտների զանգված և արտածել դրանք վարկի մարման համար անհրաժեշտ ընդհանուր գումարի չնվազման կարգով,
- ներածել `Credit` տիպի օբյեկտների զանգված և արտածել տրված `k` օր հետո առավելագույն ու նվազագույն գումարային վճարում պահանջող վարկերը:

16. Տրված է հետևյալ սահմանումը.
```
typedef enum Currency {AMD, RUB, USD, EUR,
                       GBP};
```
Իրականացնել `Deposit` դասը, որի օբյեկտները բանկային ավանդներ են:
Դաշտեր.
- ավանդի սկզբնական գումարը և տարեկան տոկոսադրույքը՝ `double` տիպի (համարել, որ տարին պարունակում է `365` օր),
- ավանդի արժույթը (`Currency` տիպի),
- ավանդի ժամկետը օրերով (`int` տիպի),
- տոկոսագումարների վերադարձման եղանակը.
  `0` - ամեն ամսվա վերջում, երբ տոկոսագումարները հաշվարկվում են ավանդի անփոփոխ մնացորդի վրա, իսկ ամսվա ընթացքում կուտակված տոկոսագումարը վերադարձվում է ավանդատուին ամեն ամսվա վերջում,
  `1` - ավանդի ժամկետի վերջում, երբ տոկոսագումարները հաշվարկվում են յուրաքանչյուր օրվա ավանդի մնացորդի վրա ու օրվա վերջում գումարվում ավանդի մնացորդին, և ընդհանուր տոկոսագումարը վերադարձվում է ավանդատուին ավանդի ժամկետի վերջում,
- `double` տիպի `4` երկարությամբ զանգված, որտեղ պահվում են արտարժույթների փոխանակման կուրսերը դրամով (պետք է լինի ստատիկ դաշտ):

Մեթոդներ և գլոբալ ֆունկցիաներ.

- կոնստրուկտոր, որը ստեղծում է տրված գումարով, տոկոսադրույքով, ժամկետով, արժույթով և տոկոսագումարների վերադարձման եղանակով ավանդ (պարամետրերին տալ լռությամբ արժեքներ),
- ավանդի սկզբնական գումարը, տոկոսադրույքը, արժույթը, ժամկետը և տոկոսագումարների վերադարձման եղանակը ստանալու և փոփոխելու ֆունկցիաներ,
- ավանդը ներածելու և արտածելու ֆունկցիաներ,
- `k` օր հետո ավանդի մնացորդը և կուտակված տոկոսագումարը տրված արտարժույթով հաշվարկող ֆունկցիաներ,
- `operator==`, `operator!=` (ավանդների համեմատման ֆունկցիաներ),
- `operator<`, `operator>` (ավանդների համեմատման ֆունկցիաներ՝ ըստ իրենց սկզբնական գումարի),
- ֆունկցիաներ՝ տրված արտարժույթի համար փոխանակնման կուրսերը ստանալու և փոփոխելու համար:

Գրել թեստ.

- նկարագրել `Deposit` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Deposit` տիպի օբյեկտների զանգված և արտածել դրանք ավանդի սկզբնական գումարի չնվազման կարգով,
- ներածել `Deposit` տիպի օբյեկտների զանգված և արտածել տրված `k` օր հետո առավելագույն և նվազագույն մնացորդ ունեցող ավանդները:

17. Տրված է հետևյալ սահմանումը.

```
class Element {
    friend class Stack;
```

```
    Element* prev;
    int data1,data2;
public:
    Element(int d1=0,int d2=0,
            Element* pr=NULL)
         :data1(d1),data2(d2),prev(pr){}
    void Show()
    { cout<<data1<<' '<<data2<<endl;}
};
```

Իրականացնել `Stack` դասը, որի օբյեկտները պահունակ են (տարրերը ամբողջ թվերի զույգեր են):

Դաշտեր.

- ցուցիչ պահունակի գագաթի վրա (`Element*` տիպի):

Մեթոդներ և գլոբալ ֆունկցիաներ.

- կոնստրուկտոր, որը ստեղծում է դատարկ պահունակ,
- պատճենման կոնստրուկտոր,
- դեստրուկտոր,
- `push` անունով ֆունկցիա, որը պահունակում ավելացնում է նոր տարր,
- `pop` անունով ֆունկցիա, որը հեռացնում է պահունակի գագաթի տարրը,
- `getd1` անունով ֆունկցիա, որը վերադարձնում է պահունակի գագաթի առաջին ամբողջ թիվը,
- `getd2` անունով ֆունկցիա, որը վերադարձնում է պահունակի գագաթի երկրորդ ամբողջ թիվը,
- `operator=` (վերագրման գործողություն),
- `operator==, operator!=` (երկու պահունակների համեմատման ֆունկցիաներ),
- պահունակն արտածելու ֆունկցիա, որը գագաթից սկսած արտածում է պահունակի պարունակությունը:

Գրել թեստ.

- նկարագրել `Stack` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- նկարագրել երկու `Stack` տիպի օբյեկտ, ներածել բնական թվեր (մինչև `0` հանդիպելը) ու ավելացնել դրանք առաջին պահունակում, երկրորդ պահունակում ավելացնել առաջին պահունակի այն տարրերը, որոնց ներսում պահվող թվերը փոխադարձաբար պարզ են, և արտածել երկրորդ պահունակը:

18. Իրականացնել `Array` դասը, որի օբյեկտները ամբողջ թվերի հաջորդականություններ են:
    Դաշտեր.
    - դինամիկ կամ ստատիկ զանգված (`int` տիպի)՝ հաջորդականության տարրերը պահելու համար,
    - հաջորդականության անդամների քանակը,
    - `Array` տիպի ստեղծված օբյեկտների քանակը (պետք է լինի ստատիկ դաշտ):

    Մեթոդներ և գլոբալ ֆունկցիաներ.
    - առանց պարամետրի կոնստրուկտոր, որը ստեղծում է դատարկ հաջորդականություն,
    - կոնստրուկտոր՝ `int` տիպի պարամետրով, որը ստեղծում է տրված քանակի անդամներով հաջորդականություն (սկզբնարժեքավորելով դրանք `0`-երով),
    - կոնստրուկտոր, որը ստեղծում է հաջորդականություն՝ ստանալով անդամները տրված `int` տիպի զանգվածից,
    - պատճենման կոնստրուկտոր,
    - դեստրուկտոր,
    - ծրագրի աշխատանքի ընթացքում ստեղծված `Array` տիպի օբյեկտների քանակը վերադարձնող ֆունկցիա,

- `operator=` (հաջորդականությունների վերագրման գործողություն),
- `operator[]` (հնարավորություն է տալիս, ըստ կարգահամարի, օգտագործել հաջորդականության անդամը՝ և՛ որպես աջակողմյան, և՛ որպես ձախակողմյան օպերանդ՝ ստուգելով կարգահամարի թույլատրելի լինելը),
- `operator>>`, `operator<<` (հաջորդականությունը ներածելու և արտածելու ֆունկցիաներ),
- `operator==`, `operator!=` (հաջորդականությունների համեմատման ֆունկցիաներ),
- հաջորդականության առավելագույն անդամը հաշվող ֆունկցիա,
- հաջորդականության պարզ տարրերի քանակը հաշվող ֆունկցիա,
- հաջորդականության վերջում նոր անդամ ավելացնելու և հաջորդականության վերջից մեկ անդամ հեռացնելու ֆունկցիաներ:

Գրել թեստ.

- նկարագրել `Array` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Array` տիպի օբյեկտ և արտածել այդ հաջորդականությունը թվանշանների քանակների աճման կարգով,
- օգտագործելով `Array` դասը՝ ներածել բնական թվերի հաջորդականություն՝ մինչև `0` հանդիպելը, և հակառակ հերթականությամբ արտածել դրա պարզ անդամները:

19. Լուծել նախորդ խնդիրը, իրականացնելով `Array` դասը որպես կաղապար դաս, ըստ հաջորդականության տարրերի տիպի:

20. Իրականացնել `Polynom` դասը, որի օբյեկտները իրական գործակիցներով բազմանդամներ են:
    Դաշտեր.
    - `double` տիպի զանգված՝ բազմանդամի գործակիցները պահելու համար,
    - բազմանդամի աստիճանը,
    - `Polynom` տիպի ստեղծված օբյեկտների քանակը:

    Մեթոդներ և գլոբալ ֆունկցիաներ.
    - առանց պարամետրի կոնստրուկտոր, որը ստեղծում է `0` գործակիցներով ու աստիճանով բազմանդամ,
    - կոնստրուկտոր՝ `int` տիպի պարամետրով, որը ստեղծում է տրված աստիճանի բազմանդամ, որի բոլոր գործակիցները հավասար են `1`-ի,
    - կոնստրուկտոր, որը ստեղծում է բազմանդամ՝ ստանալով դրա գործակիցները տրված `double` տիպի զանգվածից,
    - պատճենման կոնստրուկտոր,
    - դեստրուկտոր,
    - ծրագրի աշխատանքի ընթացքում ստեղծված `Polynom` տիպի օբյեկտների քանակը վերադարձնող ֆունկցիա,
    - `operator=` (բազմանդամների վերագրման գործողություն),
    - `operator[]` (հնարավորություն է տալիս ըստ կարգահամարի օգտագործել բազմանդամի գործակիցը և՛ որպես աջակողմյան, և՛ որպես ձախակողմյան օպերանդ՝ ստուգելով կարգահամարի թույլատրելի լինելը),
    - `operator()` (հաշվում է բազմանդամի արժեքը տրված կետում),
    - `operator>>`, `operator<<` (բազմանդամը ներածելու և արտածելու ֆունկցիաներ),

- `operator==`, `operator!=` (բազմանդամների համեմատման ֆունկցիաներ),
- `operator+`, `operator-`, `operator*`, `operator/*`, `operator%` (բազմանդամների գումարը, տարբերությունը, արտադրյալը, քանորդը և բաժանումից ստացված մնացորդը հաշվող ֆունկցիաներ),
- `operator+=`, `operator-=`, `operator*=`,`operator/=`, `operator%=` (բազմանդամների գումարը, տարբերությունը, արտադրյալը, քանորդը և բաժանումից ստացված մնացորդը ձախ օպերանդում ստացող ֆունկցիաներ),
- բազմանդամի տրված կարգի ածանցյալը հաշվող ֆունկցիա,
- բազմանդամի աստիճանը իջեցնող փակ ֆունկցիա:

Գրել թեստ.

- նկարագրել `Polynom` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Polynom` տիպի օբյեկտների զանգված և արտածել դրանց գումարն ու առաջին կարգի ածանցյալների արտադրյալը,
- ներածել `Polynom` տիպի օբյեկտների զանգված և արտածել տրված կետում մեծագույն և փոքրագույն արժեք ունեցող բազմանդամները:

21. Իրականացնել `Set` դասը, որի օբյեկտները բնական թվերի բազմություններ են:
    Դաշտեր.
    - `int` տիպի զանգված՝ բազմության տարրերը պահելու համար,
    - բազմության հզորությունը (տարրերի քանակը),
    - `Set` տիպի ստեղծված օբյեկտների քանակը:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- կոնստրուկտոր՝ `int` տիպի պարամետրով, որը ստեղծում է `1`-ից մինչև տրված թիվը բոլոր բնական թվերի բազմությունը (պարամետրին տալ լռությամբ արժեք),
- կոնստրուկտոր, որը ստեղծում է բազմություն՝ ստանալով դրա տարրերը տրված `int` տիպի զանգվածից,
- պատճենման կոնստրուկտոր,
- դեստրուկտոր,
- ծրագրի աշխատանքի ընթացքում ստեղծված `Set` տիպի օբյեկտների քանակը վերադարձնող ֆունկցիա,
- `operator=` (բազմությունների վերագրման գործողություն),
- `operator>>`, `operator<<` (բազմությունը ներածելու և արտածելու ֆունկցիաներ),
- `operator==`, `operator!=` (բազմությունների համեմատման ֆունկցիաներ),
- `operator+`, `operator-`, `operator*` (բազմությունների միավորումը, տարբերությունը և հատումը հաշվող ֆունկցիաներ),
- `operator+=`, `operator-=`, `operator*=` (բազմությունների միավորումը, տարբերությունը և հատումը ձախ օպերանդում ստացող ֆունկցիաներ),
- բազմության հզորությունը վերադարձնող ֆունկցիա,
- բազմությանը տարր ավելացնելու ֆունկցիա,
- ֆունկցիա, որը ստուգում է տրված թվի պատկանելիությունը բազմությանը,
- բազմությունից տրված թիվը հեռացնելու ֆունկցիա, որը նաև պարզում է՝ արդյո՞ք տրված թիվը պատկանում էր բազմությանը, թե ոչ,

- ֆունկցիա, որը ստուգում է բազմությունների՝ մեկը մյուսի ենթաբազմություն լինելը,
- բազմության մեծագույն տարրը վերադարձնող ֆունկցիա,
- բազմությունը կարգավորող փակ ֆունկցիա:

Գրել թեստ.

- նկարագրել `Set` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Set` տիպի օբյեկտների զանգված և արտածել բոլոր բազմությունների հատումն ու միավորումը,
- ներածել `Set` տիպի օբյեկտների զանգված և արտածել այն բազմությունը, որը առավելագույն թվով բազմությունների ենթաբազմություն,
- ներածել բնական թվերի հաջորդականություն՝ մինչև `0` հանդիպելը, և արտածել այդ հաջորդականության իրարից տարբեր տարրերի քանակը (օգտագործել `Set` դասը):

22. Իրականացնել `Vector` դասը, որի օբյեկտները $n \geq 2$ չափանի տարածության վեկտորներ են:

Դաշտեր.

- միաչափ զանգված (`double` տիպի)՝ վեկտորի կոորդինատները պահելու համար,
- վեկտորի չափողականությունը:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- կոնստրուկտոր՝ `int` տիպի պարամետրով, որը ստեղծում է տրված չափողականությամբ զրոյական վեկտոր, և այն հնարավոր չէ օգտագործել `int` տիպից դեպի `Vector` տիպ անբացահայտ ձևափոխության համար (պարամետրին տալ լռությամբ արժեք),

- կոնստրուկտոր, որը ստեղծում է տրված չափողականությամբ վեկտոր՝ ստանալով կոորդինատները տրված `double` տիպի զանգվածից,
- պատճենման կոնստրուկտոր,
- դեստրուկտոր,
- `operator=` (վեկտորների վերագրման գործողություն),
- `operator[]` (հնարավորություն է տալիս ըստ կարգահամարի օգտագործել վեկտորի կոորդինատը և՛ որպես աջակողմյան, և՛ որպես ձախակողմյան օպերանդ՝ ստուգելով կարգահամարի թույլատրելի լինելը),
- `operator>>`,`operator<<` (վեկտորը ներածելու և արտածելու ֆունկցիաներ),
- երկու վեկտորների այբբենական կարգով համեմատման ֆունկցիաներ՝ `operator==`, `operator!=`, `operator<`, `operator<=`, `operator>`, `operator>=`,
- `operator+`, `operator-` (վեկտորների գումարը և տարբերությունը հաշվող ֆունկցիաներ),
- `operator+=`, `operator-=` (վեկտորների գումարը և տարբերությունը ձախ օպերանդում ստացող ֆունկցիաներ),
- `operator*` (վեկտորների սկալյար արտադրյալը հաշվող ֆունկցիա),
- երկու վեկտորների միջև հեռավորությունը հաշվող ֆունկցիա:

Գրել թեստ.
- նկարագրել `Vector` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել տրված չափողականությամբ `Vector` տիպի օբյեկտների զանգված, արտածել իրարից ամենահեռու

վեկտորները և կոորդինատների սկզբնակետին ամենամոտ վեկտորը:

23. Իրականացնել `String` դասը, որի օբյեկտները տողեր են:
Դաշտեր.
- միաչափ զանգված (`char` տիպի)՝ տողի սիմվոլները պահելու համար,
- տողի սիմվոլների քանակը:

Մեթոդներ և գլոբալ ֆունկցիաներ.
- առանց պարամետրի կոնստրուկտոր, որը ստեղծում է դատարկ տող,
- կոնստրուկտոր, որը ստեղծում է տող՝ ստանալով սիմվոլները տրված `char` տիպի զանգվածից,
- պատճենման կոնստրուկտոր,
- դեստրուկտոր,
- `operator=` (տողերի վերագրման գործողություն),
- `operator[]` (հնարավորություն է տալիս օգտագործել տողի համապատասխան կարգահամարով սիմվոլը և՛ որպես աջակողմյան, և՛ որպես ձախակողմյան օպերանդ՝ ստուգելով կարգահամարի թույլատրելի լինելը),
- `operator>>`, `operator<<` (տողը ներածելու և արտածելու ֆունկցիաներ),
- `operator+` (երկու տողերը կցելու ֆունկցիա),
- `operator+=` (երկու տողերը կցելու և արդյունքը ձախ օպերանդում ստանալու ֆունկցիա),
- տողերը այբբենական կարգով համեմատելու ֆունկցիաներ՝ `operator==`, `operator!=`, `operator<`, `operator<=`, `operator>`, `operator>=`,
- տողի երկարությունը վերադարձնող ֆունկցիա,
- տողում տրված ենթատողը փնտրելու ֆունկցիա,

- տողում տրված ենթատողը մեկ այլ տողով փոխարինելու ֆունկցիա,
- տողից ենթատողի ստացման ֆունկցիա՝ ըստ ենթատողի սկզբի կարգահամարի և երկարության:

Գրել թեստ.

- նկարագրել `String` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `String` տիպի օբյեկտների զանգված և այբբենական կարգով արտածել զանգվածի տարրերը,
- ներածել `String` տիպի օբյեկտների զանգված և արտածել դրանց բոլորի ամենաերկար ընդհանուր նախածանցը:

24. Իրականացնել `Matrix` դասը, որի օբյեկտները իրական թվերի մատրիցներ են:

Դաշտեր.

- միաչափ զանգված կամ դինամիկ երկչափ զանգված (`double` տիպի)՝ մատրիցի տարրերը պահելու համար,
- մատրիցի տողերի և սյուների քանակները:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- կոնստրուկտոր՝ երկու `int` տիպի պարամետրով, որը ստեղծում է տրված քանակի տողերով և սյուներով մատրից, որի բոլոր տարրերը հավասար են 0-ի (պարամետրերին տալ լռությամբ արժեքներ),
- կոնստրուկտոր, որը ստեղծում է տրված քանակի տողերով և սյուներով մատրից՝ ստանալով տարրերը՝ տրված `double` տիպի դինամիկ երկչափ զանգվածից,
- պատճենման կոնստրուկտոր,
- դեստրուկտոր,
- `operator=` (մատրիցների վերագրման գործողություն),

- `operator>>`, `operator<<` (մատրիցը ներածելու և արտածելու ֆունկցիաներ),
- `operator==`,`operator!=` (մատրիցների համեմատման ֆունկցիաներ),
- `operator+`, `operator-`, `operator*` (մատրիցների գումարը, տարբերությունը և արտադրյալը հաշվող ֆունկցիաներ),
- `operator+=`, `operator-=`, `operator*=` (մատրիցների գումարը, տարբերությունը և արտադրյալը ձախ օպերանդում ստացող ֆունկցիաներ),
- `operator()` (հնարավորություն է տալիս ըստ տրված կարգահամարների օգտագործել մատրիցի տարրը և՛ որպես աջակողմյան, և՛ որպես ձախակողմյան օպերանդ՝ ստուգելով կարգահամարների թույլատրելի լինելը),
- մատրիցի տողերի և սյուների քանակը ստացող և փոփոխող ֆունկցիաներ,
- մատրիցի երկու տողը կամ երկու սյունը տեղերով փոխելու ֆունկցիաներ:

Գրել թեստ.

- նկարագրել `Matrix` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Matrix` տիպի օբյեկտ, դասավորել կենտ համարով տողերի տարրերը չնվազման կարգով և արտածել մատրիցը,
- ներածել `Matrix` տիպի օբյեկտ, դասավորել մատրիցի տողերը՝ դրանց տարրերի գումարների չնվազման կարգով և արտածել մատրիցը:

25. Լուծել նախորդ խնդիրը՝ իրականացնելով `Matrix` դասը՝ որպես կաղապար դաս՝ ըստ մատրիցի տարրերի տիպի:

26. Իրականացնել `Decimal` դասը, որի օբյեկտները ֆիքսված ստորակետով թվեր են, որոնք ստորակետից առաջ և հետո ունեն առավելագույնը 9 նիշ:
    Մեթոդներ և գլոբալ ֆունկցիաներ.
    - կոնստրուկտոր՝ `int` տիպի պարամետրով, որը ստեղծում է տրված ամբողջ թվին հավասար ֆիքսված ստորակետով թիվ (պարամետրին տալ լռությամբ `0` արժեք),
    - կոնստրուկտոր՝ `double` տիպի պարամետրով, որը ստեղծում է տրված սահող ստորակետով թվին հնարավորինս մոտ ֆիքսված ստորակետով թիվ,
    - ֆիքսված ստորակետով թվերի համեմատման ֆունկցիաներ՝ `operator==`, `operator!=`, `operator<`, `operator<=`, `operator>`, `operator>=`,
    - `operator+`, `operator-`, `operator*`, `operator/` (ֆիքսված ստորակետով թվերի գումարը, տարբերությունը, արտադրյալը և հարաբերությունը հաշվող ֆունկցիաներ),
    - `operator+=`, `operator-=`, `operator*=`, `operator/=` (ֆիքսված ստորակետով թվերի գումարը, տարբերությունը, արտադրյալը և հարաբերությունը. ձախ օպերանդում ստացող ֆունկցիաներ),
    - `operator++`, `operator--` (ինկրեմենտի ու դեկրեմենտի պոստֆիքս ու պրեֆիքս տարբերակները),
    - `operator>>`, `operator<<` (ֆիքսված ստորակետով թիվը ներածելու և արտածելու ֆունկցիաներ),
    - `operator int`, `operator double` (դեպի `int` և դեպի `double` տիպի ձևափոխություն իրականացնող ֆունկցիաներ):

Գրել թեստ.

- նկարագրել `Decimal` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `Decimal` տիպի օբյեկտների զանգված և արտածել այն թվերը, որոնք մեծ են տրված թվից,
- ներածել `Decimal` տիպի օբյեկտների զանգված և արտածել դրանց գումարն ու արտադրյալը,
- ներածել `Decimal` տիպի օբյեկտների զանգված և արտածել դրանցից մեծագույնը և փոքրագույնը:

27. Իրականացնել `DateTime` դասը, որի օբյեկտները իրենցից ներկայացնում են ամսաթիվ և ժամ (վայրկյանի ճշտությամբ):
Մեթոդներ և գլոբալ ֆունկցիաներ.
    - կոնստրուկտոր՝ `int` տիպի պարամետրերով, որը ստեղծում է ամսաթիվ և ժամ՝ օգտագործելով տրված տարին, ամիսը, օրը, ժամը, րոպեն և վայրկյանը (պարամետրերին տալ լռությամբ արժեքներ),
    - ամսաթվի և ժամի որևէ բաղադրիչը (տարին, ամիսը, օրը, ժամը, րոպեն կամ վայրկյանը) ստացող և փոփոխող ֆունկցիաներ,
    - շաբաթվա օրը վերադարձնող ֆունկցիա,
    - `operator-` (երկու օբյեկտների միջև ժամանակահատվածը վայրկյաններով ստացող ֆունկցիա),
    - ամսաթվին և ժամին տրված քանակով տարի/ամիս/օր/ժամ/րոպե/վայրկյան ավելացնելու ֆուկցիաներ,
    - երկու օբյեկտների համեմատման ֆունկցիաներ՝ `operator==`, `operator!=`, `operator<`, `operator<=`, `operator>`, `operator>=`,
    - `operator>>`, `operator<<` (ամսաթիվը և ժամը ներածելու և արտածելու ֆունկցիաներ):

Գրել թեստ.

- նկարագրել `DateTime` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `DateTime` տիպի օբյեկտների զանգված և արտածել դրանցից մեծագույնի և փոքրագույնի միջև ընկած ժամանակահատվածը՝ վայրկյաններով,
- ներածել `DateTime` տիպի օբյեկտների զանգված և չնվազման կարգով արտածել զանգվածի այն տարրերը, որոնց համապատասխան շաբաթվա օրը երկուշաբթի է:

28. Իրականացնել `MultiSet` դասը, որի օբյեկտները `[0,10000]` միջակայքի ամբողջ թվերի մուլտիբազմություններ են:
Մեթոդներ և գլոբալ ֆունկցիաներ.
    - առանց պարամետրի կոնստրուկտոր, որը ստեղծում է դատարկ մուլտիբազմություն,
    - կոնստրուկտոր, որը ստեղծում է մուլտիբազմություն՝ ստանալով դրա տարրերը՝ տրված `int` տիպի զանգվածից,
    - պատճենման կոնստրուկտոր,
    - դեստրուկտոր,
    - `operator=` (մուլտիբազմությունների վերագրման գործողություն),
    - `operator>>`, `operator<<` (մուլտիբազմությունը ներածելու և արտածելու ֆունկցիաներ),
    - `operator==`, `operator!=` (մուլտիբազմությունների համեմատման ֆունկցիաներ),
    - `operator|`, `operator-`, `operator&` (մուլտիբազմությունների միավորումը, տարբերությունը և հատումը հաշվող ֆունկցիաներ),

- `operator|=, operator-=, operator&=` (մուլտիբազմությունների միավորումը, տարբերությունը և հատումը ձախ օպերանդում ստացող ֆունկցիաներ),
- տրված ամբողջ թիվը մուլտիբազմությանը ավելացնելու (հանդիպումների քանակի `1`-ով մեծացնելու) ֆունկցիա, որը վերադարձնում է ավելացնելուց հետո տրված ամբողջ թվի հանդիպումների քանակը,
- մուլտիբազմությունից տրված ամբողջ թիվը հեռացնելու (հանդիպումների քանակի `1`-ով փոքրացնելու) ֆունկցիա, որը վերադարձնում է հեռացնելուց հետո տրված ամբողջ թվի հանդիպումների քանակը,
- ֆունկցիա, որը վերադարձնում է տրված ամբողջ թվի հանդիպումների քանակը մուլտիբազմության մեջ,
- ֆունկցիա, որը ստուգում է մուլտիբազմությունների՝ մեկը մյուսի ենթաբազմություն լինելը:

Գրել թեստ.

- նկարագրել `MultiSet` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `MultiSet` տիպի օբյեկտների զանգված և արտածել բոլորի հատումն ու միավորումը,
- ներածել `MultiSet` տիպի օբյեկտների զանգված և արտածել առավելագույն (նվազագույն) ամբողջ թիվը պարունակող մուլտիբազմությունը,
- ներածել բնական թվերի հաջորդականություն՝ մինչև `0` հանդիպելը և արտածել այդ հաջորդականության ամենաշատ հանդիպող անդամը (օգտագործել `MultiSet` դասը):

29. Իրականացնել `BitVector` դասը, որի օբյեկտները բիթերի հաջորդականություններ են:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- կոնստրուկտոր՝ `int` տիպի պարամետրով, որը ստեղծում է տրված քանակով զրոյական բիթերից կազմված բիթային վեկտոր և այն հնարավոր չէ օգտագործել `int` տիպից դեպի `BitVector` տիպ անբացահայտ ձևափոխության համար (պարամետրին տալ լռությամբ արժեք),
- կոնստրուկտոր, որը ստեղծում է բիթային վեկտոր՝ ստանալով բիթերը տրված `bool` տիպի զանգվածից,
- պատճենման կոնստրուկտոր,
- դեստրուկտոր,
- `operator=` (բիթային վեկտորների վերագրման գործողություն),
- `operator[]` (հնարավորություն է տալիս օգտագործել վեկտորի տրված կարգահամարով բիթը և՛ որպես աջակողմյան, և՛ որպես ձախակողմյան օպերանդ՝ ստուգելով կարգահամարի թույլատրելի լինելը),
- `operator>>`, `operator<<` (բիթային վեկտորը ներածելու և արտածելու ֆունկցիաներ),
- `operator==`, `operator!=` (երկու բիթային վեկտորների համեմատման ֆունկցիաներ),
- `operator|`, `operator&` (բիթային վեկտորների բիթային «կամ» ու բիթային «և» գործողության արդյունքը հաշվող ֆունկցիաներ),
- `operator|=`, `operator&=` (բիթային վեկտորների բիթային «կամ» ու բիթային «և» գործողության արդյունքը ձախ օպերանդում ստացող ֆունկցիաներ),
- `operator+` (երկու բիթային վեկտորները կցելու ֆունկցիա),
- `operator+=` (երկու բիթային վեկտորները կցելու և արդյունքը ձախ օպերանդում ստանալու ֆունկցիա),

- `operator>>`, `operator<<` (բիթային վեկտորի աջ և ձախ տեղաշարժն իրականացնող ֆունկցիաներ),
- `operator>>=`, `operator<<=` (բիթային վեկտորի աջ և ձախ տեղաշարժն իրականացնող ու արդյունքը ձախ օպերանդում ստացող ֆունկցիաներ):
- բիթային վեկտորի բիթերի քանակը վերադարձնող ֆունկցիա:

Գրել թեստ.

- նկարագրել `BitVector` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել `BitVector` տիպի օբյեկտների զանգված և արտածել բոլորի բիթային «կամ» ու բիթային «և» գործողության արդյունքում ստացված վեկտորը:

30. Իրականացնել `BigInteger` դասը, որի օբյեկտները առավելագույնը `100000` թվանշան պարունակող ամբողջ թվեր են:
Մեթոդներ և գլոբալ ֆունկցիաներ.
    - կոնստրուկտոր՝ `int` տիպի պարամետրով, որը ստեղծում է տրված ամբողջ թվին հավասար օբյեկտ (պարամետրին տալ լռությամբ `0` արժեք),
    - կոնստրուկտոր, որը ստեղծում է տրված նշանով ամբողջ թիվ՝ ստանալով թվանշանները տրված `int` տիպի զանգվածից (նշանը ներկայացնող պարամետրի համար սահմանել դրական նշանը որպես լռությամբ արժեք),
    - պատճենման կոնստրուկտոր,
    - դեստրուկտոր,
    - `operator=` (ամբողջ թվերի վերագրման գործողություն),
    - `operator>>`, `operator<<` (ամբողջ թիվը ներածելու և արտածելու ֆունկցիաներ),

- ամբողջ թվերի համեմատման ֆունկցիաներ՝ `operator==`, `operator!=`, `operator<`, `operator<=`, `operator>`, `operator>=`,
- `operator+`, `operator-`, `operator*`, `operator/`*, `operator%` (ամբողջ թվերի գումարը, տարբերությունը, արտադրյալը, քանորդը և բաժանումից ստացված մնացորդը հաշվող ֆունկցիաներ),
- `operator+=`, `operator-=`, `operator*=`,`operator/=`, `operator%=` (ամբողջ թվերի գումարը, տարբերությունը, արտադրյալը, քանորդը և բաժանումից ստացված մնացորդը ձախ օպերանդում ստացող ֆունկցիաներ),
- `operator++`, `operator--` (ինկրեմենտի ու դեկրեմենտի պոստֆիքս ու պրեֆիքս տարբերակները),
- `operator^` (ամբողջ թվի տրված ամբողջ աստիճանը հաշվող ֆունկցիա),
- `operator int` (դեպի `int` տիպը ձևափոխություն իրականացնող ֆունկցիա):

Գրել թեստ.

- նկարագրել `BigInteger` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- արտածել տրված բնական թվի ֆակտորիալը՝ օգտագործելով `BigInteger` դասը,
- ներածել `BigInteger` տիպի օբյեկտների զանգված և արտածել դրանց գումարը, արտադրյալը և առավելագույն տարրը:

31. Իրականացնել `List` դասը, որի օբյեկտները երկկապ կապակցված ցուցակներ են (յուրաքանչյուր հանգույցում պահվում է նախորդ և հաջորդ հանգույցների հասցեները): `List`

դասը պետք է սահմանել որպես կաղապար դաս՝ ըստ հանգույցներում պահվող ինֆորմացիայի տիպի:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- առանց պարամետրի կոնստրուկտոր, որը ստեղծում է դատարկ ցուցակ,
- կոնստրուկտոր՝ `int` տիպի պարամետրով, որը ստեղծում է տրված քանակի հանգույցներով կապակցված ցուցակ (հանգույցների ինֆորմացիայի լռելյայն արժեքներով), և այն հնարավոր չէ օգտագործել `int` տիպից դեպի `List` տիպ անբացահայտ ձևափոխության համար,
- կոնստրուկտոր, որը ստեղծում է տրված քանակի հանգույցներով կապակցված ցուցակ՝ ստանալով հանգույցների ինֆորմացիան տրված զանգվածից,
- պատճենման կոնստրուկտոր,
- դեստրուկտոր,
- `operator=` (ցուցակների վերագրման գործողություն),
- `operator[]` (հնարավորություն է տալիս օգտագործել ցուցակի տրված կարգահամարով հանգույցում պահվող ինֆորմացիան և՛ որպես աջակողմյան, և՛ որպես ձախակողմյան օպերանդ՝ ստուգելով կարգահամարի թույլատրելի լինելը),
- `operator>>`, `operator<<` (կապակցված ցուցակը ներածելու և արտածելու ֆունկցիաներ),
- `operator==`, `operator!=` (երկու ցուցակների համեմատման ֆունկցիաներ),
- `operator+` (երկու կապակցված ցուցակներն իրար կցող ֆունկցիա),
- `operator+=` (երկու կապակցված ցուցակներն իրար կցող և արդյունքը ձախ օպերանդում ստացող ֆունկցիա),

- կապակցված ցուցակից ցուցիչով տրված հանգույցի հեռացման ֆունկցիա,
- կապակցված ցուցակում ցուցիչով տրված հանգույցից հետո տրված ինֆորմացիայով նոր հանգույցի ավելացման ֆունկցիա,
- կապակցված ցուցակի սկզբից (վերջից) տրված ինֆորմացիայով նոր հանգույցի ավելացման ֆունկցիաներ,
- կապակցված ցուցակի սկզբից (վերջից) հանգույցի հեռացման ֆունկցիաներ,
- կապակցված ցուցակի առաջին (վերջին) հանգույցների հասցեները ստացող ֆունկցիաներ,
- կապակցված ցուցակի հանգույցների քանակը վերադարձնող ֆունկցիա:

Գրել թեստ.

- նկարագրել `List` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- ներածել բնական թվեր պարունակող կապակցված ցուցակներ, կարգավորել դրանք չնվազման կարգով և կցել այդ ցուցակները՝ ստանալով մեկ կարգավորված (չնվազման կարգով) կապակցված ցուցակ:

11. Ժառանգում և պոլիմորֆիզմ

Ժառանգումը օբյեկտ-կողմնորոշված ծրագրավորման հիմնական սկզբունքներից մեկն է: Ենթադրենք անհրաժեշտ է իրականանցնել `A` դասը, որի օբյեկտները հանդիսանում են նաև `B` դասի օբյեկտներ՝ որոշ լրացուցիչ հնարավորություններով: Այդ դեպքում `B` դասի ֆունկցիոնալությունը կրկին `A` դասում իրականացնելու փոխարեն պետք է պարզապես `A` դասը ժառանգել `B` դասից՝ խուսափելով կոդի անտեղի կրկնությունից, որն էլ իր հերթին փոքրացնում է ծրագրում սխալներ թույլ տալու հավանականությունը և նպաստում է ավելի հեշտ ուղեկցվող կոդի ստացմանը: C++ լեզվում `A` դասը `B` դասից ժառանգելու համար `A` դասի նկարագրությունը պետք է ունենա հետևյալ տեսքը՝

```
class A:[տեսանելիություն]B{
    դասի մարմին
};
```

դասի մարմին-ը `A` դասի, իսկ *տեսանելիություն*-ը ժառանգման տեսակն է, որը կարող է լինել `public`, `protected` և `private`: Եթե *տեսանելիություն*-ը բացակայում է, ապա ժառանգման տեսակը լինում է `private`: Հիմնականում օգտագործվում է ժառանգման `public` տեսակը: `B` դասը կոչվում է բազային դաս, իսկ `A` դասը՝ ժառանգ դաս: Ժառանգման արդյունքում բազային դասի բոլոր անդամները դառնում են անդամներ նաև ժառանգ դասի համար: Ընդ որում ժառանգված անդամների տեսանելիությունը կախված է ժառանգման տեսակից և որոշվում է ըստ հետևյալ աղյուսակի՝

Ժառանգման տեսակ / Բազային դասի անդամի տեսանելիություն	public	protected	private
public	public	protected	private
protected	protected	protected	private
private	private	private	private

Քանի որ ժառանգ դասի օբյեկտները հանդիսանում են նաև բազային դասի օբյեկտներ՝ պարունակելով բազային դասի բոլոր անդամները, ապա ծրագրում պետք է հնարավորություն լինի անհրաժեշտության դեպքում ժառանգ դասի օբյեկտները դիտարկել որպես բազային դասի օբյեկտներ: Դա անելու համար թույլատրվում է կատարել տիպերի ձևափոխություն (բացահայտ կամ անբացահայտ)՝ ժառանգ դասի օբյեկտի հղումից (ցուցիչից) դեպի բազային դասի օբյեկտի հղում (ցուցիչ)՝

```
B b;
A a;
/*b1-ը բազային դասի օբյեկտի հղում է, որն իրականում
ժառանգ դասի օբյեկտ է*/
B& b1=a;
/*ptr-ն բազային դասի օբյեկտի հասցե է, որը ցույց է տալիս
ժառանգ դասի օբյեկտի վրա*/
B* ptr=&a;
```

Քանի որ բազային դասի հղումը (ցուցիչը) կարող է նաև ցույց տալ բազային դասի օբյեկտի վրա, ապա հակառակ ձևափոխությունը ծրագրի աշխատանքի ընթաքում կարող է բերել սխալների: Դա է պատճառը, որ հակառակ ձևափոխությունը թույլատրվում է կատարել միայն բացահայտ տարբերակով՝

```
B b;
A a;
B* ptr;
A* ptr1;
ptr=&a;
/*այս ձևափոխությունն անվտանգ է, քանի որ ptr-ն ցույց է տալիս ժառանգ դասի օբյեկտի վրա*/
ptr1=(A*)ptr;
ptr=&b;
/*այս ձևափոխությունը վտանգավոր է, քանի որ ptr-ն ցույց չի տալիս ժառանգ դասի օբյեկտի վրա և ptr1-ի միջոցով ժառանգ դասի անդամներին դիմելիս կլինեն անկանխատեսելի հետևանքներ*/
ptr1=(A*)ptr;
```

C++ լեզվում թույլատրվում է բազմակի ժառանգումը, երբ ժառանգ դասը ժառանգվում է մեկից ավելի բազային դասերից՝

```
class A:[տեսանելիություն1]B1,…,[տեսանելիությունn]Bn{
    դասի մարմին
};
```

n-ը բազային դասերի քանակն է՝ $n \geq 1$: Նշենք նաև, որ ժառանգ դասի կոնստրուկտորների սկզբնարժեքավորման ցուցակում պարտադիր պետք է լինի բազային դասի մեկ կոնստրուկտորի կանչ՝ ժառանգ դասի օբյեկտը ստեղծելիս բազային դասի ենթաօբյեկտը ստեղծելու համար: Եթե այդ կանչը բացակայում է, ապա կոմպիլյատորը ավելացնում է բազային դասի լռությամբ կոնստրուկտորի կանչը, եթե այն գոյություն ունի (եթե բազային դասում լռությամբ կոնստրուկտոր չկա կամ տեսանելի չէ, ապա կոմպիլյացիան կավարտվի սխալներով): Այժմ լուծենք հետևյալ խնդիրը՝

Օրինակ 1. Ուղղել հետևյալ ծրագրի բոլոր սխալները և նշել, թե ինչ կարտածի այն սխալներն ուղղելուց հետո.

```
class Time{
    int hour,min;
protected:
    void print(){ cout<<hour<<':'<<min; }
public:
    Time(int,int);
};
class LTime:public Time{
    int sec;
public:
    LTime(int h,int m,int s)
        :Time(h,m) {
            sec=s;
            cout<<"Time from constructor ";
            Time::print();
            cout<<':'<<sec<<endl;
    }
};
int main(){
    Time t(2,48);
    t.print(); cout<<endl;
    LTime lt(10,35,50);
    cout<<"The same time from main ";
    lt.print(); cout<<endl;
    return 0;
}
```

Լուծում՝ ծրագրում առկա է հետևյալ քերականական սխալը. քանի որ `Time` դասում հայտարարված `print` մեթոդն ունի `protected` տեսանելիություն, ապա այն օգտվելու համար հասանելի չէ `main` ֆունկցիայից: Սխալն ուղղելու համար անհրաժեշտ է փոփոխել այդ մեթոդի `protected` տեսանելիության տիրույթը `public` տեսանելիության տիրույթով: Բացի վերը նշված քերականական սխալից, ծրագրում կա ևս մեկ սխալ, որն ի հայտ

է գալիս կապակցման (linking) փուլում. Time դասում հայտարարված երկու պարամետրանոց կոնստրուկտորը օգտագործվում է ծրագրում, սակայն իրականացված չէ: Սխալն ուղղելու համար անհրաժեշտ է իրականացնել այն դասի ներսում կամ դասից դուրս: Դասի ներսում իրականացնելու համար կարող ենք Time(int,int) տողը փոխարինել Time(int h,int m) :hour(h),min(m){} տողով: Նշված եղանակով սխալներն ուղղելուց հետո ծրագիրը կարտածի հետևյալը՝

```
2:48
Time from constructor 10:35:50
The same time from main 10:35
```

Այժմ անդրադառնանք օբյեկտ-կողմնորոշված ծրագրավորման հաջորդ հիմնական սկզբունքին՝ պոլիմորֆիզմին: Հաճախ կարիք է լինում այնպես անել, որ բազային դասի մեթոդը այլ գործողություններ կատարի այն դեպքում, երբ կանչվում է ժառանգ դասի օբյեկտի համար: Դա անելու համար կարող ենք ժառանգ դասում նկարագրել և իրականացնել բազային դասի այդ նույն մեթոդը՝

```
class B{
public:
    void Print(){
        cout<<"This is base class member.\n";
    }
};

class A:public B{
public:
    void Print(){
        cout<<"This is derived class member.\n";
    }
};
```

Այս տարբերակի թերությունն այն է, որ երբ կանչենք համապատասխան մեթոդը՝ օգտագործելով բազային դասի հղում (ցուցիչ), ապա կանչվելու է բազային դասի մեթոդը նույնիսկ այն դեպքում, երբ հղումը (ցուցիչը) իրականում ցույց է տալիս ժառանգ դասի օբյեկտի վրա՝

```
int main(){
    A a;
    //կանչվում է A դասի Print մեթոդը
    a.Print();
    B* ptr=&a;
    /*կանչվում է B դասի Print մեթոդը, չնայած, որ ptr-ն A
    դասի օբյեկտի հասցե է*/
    ptr->Print();
    return 0;
}
```

Որպեսզի խուսափենք վերը նշված թերությունից, անհրաժեշտ է համապատասխան մեթոդը հայտարարել որպես վիրտուալ ֆունկցիա՝ դիմացից ավելացնելով `virtual` ծառայողական բառը: Եթե դասի մեթոդը վիրտուալ է, և այն կանչում ենք՝ օգտագործելով հղում (ցուցիչ), ապա կանչվում է այն դասի մեթոդը, որի օբյեկտի վրա ցույց է տալիս հղումը (ցուցիչը): Ինչպես տեսանք, ոչ վիրտուալ մեթոդի դեպքում կանչվում է այն դասի մեթոդը, որի տիպն ունի հղումը (ցուցիչը): Չնափոխենք վերևի օրինակը՝

```
class B{
public:
    virtual void Print(){
        cout<<"This is base class member.\n";
    }
};
```

```
class A:public B{
public:
     [virtual] void Print() [override]{
        cout<<"This is derived class member.\n";
    }
};
int main(){
    A a;
    B b;
    B* ptr;
    ptr=&b;
    //կանչվում է B դասի Print մեթոդը
    ptr->Print();
    ptr=&a;
    //կանչվում է A դասի Print մեթոդը
    ptr->Print();
    return 0;
}
```

Ժառանգ դասում վիրտուալ ֆունկցիան վերասահմանելիս, `virtual` և `override` ծառայողական բառերը նշելը պարտադիր չեն: Ինչպես տեսնում ենք այս օրինակում, `Print` ֆունկցիայի միատեսակ երկու կանչերի դեպքում՝ `ptr->Print()`, կանչվում են տարբեր մեթոդներ՝ կախված նրանից, թե ինչ տիպի օբյեկտի վրա է ցույց տալիս `ptr` ցուցիչը: Դա է պատճառը, որ վիրտուալ ֆունկցիաները կոչվում են նաև պոլիմորֆ (բազմաձև) ֆունկցիաներ: Պոլիմորֆիզմի այս տեսակը կոչվում է նաև դինամիկ պոլիմորֆիզմ, քանի որ ծրագրի աշխատանքի ընթացքում է որոշվում, թե որ դասի վիրտուալ մեթոդը պետք է կանչվի: Վիրտուալ մեթոդը վերասահմանելիս կարելի է նաև կանչել բազային դասի համապատասխան մեթոդը՝ մեթոդի անունից առաջ նշելով համապատասխան դասի անունը և `::`՝

```
class A:public B{
```

```
public:
    virtual void Print() override{
        /*կանչվում ենք B դասի Print մեթոդը ընթացիկ
        օբյեկտի համար*/
        B::Print();
        cout<<"This is derived class member.\n";
    }
};
```

Լինում են դեպքեր, երբ բազային դասը նախատեսված է միայն ժառանգվելու համար, այդ դասի օբյեկտներ չեն ստեղծվելու և իմաստ չի ունենում այդ դասում նկարագրված վիրտուալ մեթոդները իրականացնելը: Այդ դեպքերի համար հնարավորություն է տրվում չիրականացնել վիրտուալ մեթոդը՝ սահմանելով այն որպես մաքուր վիրտուալ (աբստրակտ) մեթոդ, որի համար պետք է պարզապես մեթոդի հայտարարությունից հետո ավելացնել =0՝

```
class Shape{
public:
    virtual void Print()=0;
};

class Rectangle:public Shape{
public:
    virtual void Print() override
    {
        cout<<"This is Rectangle.\n";
    }
};
```

Այն դասերը, որոնք պարունակում են գոնե մեկ մաքուր վիրտուալ մեթոդ, կոչվում են աբստրակտ դասեր: Արգելվում է ստեղծել աբստրակտ դասի օբյեկտ: Եթե ժառանգ դասում վերասահ-

մանված չէ բազային դասի որևէ աբստրակտ մեթոդ, ապա ժառանգ դասում այդ մեթոդը մնում է մաքուր վիրտուալ, իսկ ժառանգ դասը բազային դասի նման լինում է աբստրակտ դաս: Օգտագործելով ժառանգումն ու պոլիմորֆիզմը՝ լուծենք հետևյալ խնդիրը՝

Օրինակ 2. Իրականացնել հետևյալ դասերը՝ `Shape`, `Triangle`, `Rectangle`, `Circle`

- `Shape` դասը հանդիսանում է բազային աբստրակտ դաս մնացած դասերի համար, և դրանից ժառանգված դասերի օբյեկտներն իրենցից ներկայացնում են երկրաչափական պատկերներ,
- `Triangle` դասի օբյեկտները եռանկյուններ են,
- `Rectangle` դասի օբյեկտներն ուղղանկյուններ են,
- `Circle` դասի օբյեկտները շրջաններ են:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- `Shape` դասից ժառանգված բոլոր դասերը պետք է պարունակեն տվյալ դասի օբյեկտ ստեղծելու համար անհրաժեշտ կոնստրուկտոր(ներ) կամ մեթոդներ,
- պատկերի պարագիծը և մակերեսը հաշվող ֆունկցիաներ, որոնք `Shape` դասում պետք է սահմանված լինեն որպես մաքուր վիրտուալ ֆունկցիաներ,
- պատկերն արտածող ֆունկցիա, որը `Shape` դասում պետք է սահմանված լինի որպես մաքուր վիրտուալ ֆունկցիա:

Գրել թեստ.

- կառուցել պատկերների զանգված (`Shape*` տիպի զանգված) և արտածել բոլորի պարագծերը,
- կառուցել պատկերների զանգված (`Shape*` տիպի զանգված) և արտածել դրանք մակերեսների չնվազման կարգով:

Լուծում՝

```
#include <iostream>
```

```
#include <cmath>

using std::cout;
using std::endl;
using std::swap;

class Shape
{
public:
    //հաշվում է մակերեսը
    virtual double Area() const=0;
    //հաշվում է պարագիծը
    virtual double Perimeter() const=0;
    //արտածում է ինֆորմացիա պատկերի մասին
    virtual void Print() const=0;
};

class Triangle:public Shape
{
public:
    //ստեղծում է տրված կողմերով եռանկյուն
    Triangle(double a=0,double b=0,double c=0);
    virtual double Area() const;
    virtual double Perimeter() const
    { return first+second+third; }
    virtual void Print() const {
        cout<<"Triangle("<<first<<','
            <<second<<','<<third<<')';
    }
private:
    double first; //առաջին կողմի երկարությունը
    double second; //երկրորդ կողմի երկարությունը
    double third; //երրորդ կողմի երկարությունը
};
```

```
class Rectangle:public Shape
{
public:
    //ստեղծում է տրված կողմերով ուղղանկյուն
    Rectangle(double w=0,double l=0)
        :width(w>=0?w:-w),length(l>=0?l:-l){}
    virtual double Area() const
    { return width*length; }
    virtual double Perimeter() const
    { return 2*(width+length); }
    virtual void Print() const {
        cout<<"Rectangle("<<width
            <<","<<length<<')';
    }
private:
    double width; //լայնությունը
    double length; //երկարությունը
};

class Circle:public Shape
{
public:
    //ստեղծում է տրված շառավղով շրջան
    Circle(double r=0)
        :radius(r>=0?r:-r){}
    virtual double Area() const
    { return pi*radius*radius; }
    virtual double Perimeter() const
    { return 2*pi*radius; }
    virtual void Print() const
    { cout<<"Circle("<<radius<<')'; }
private:
    const static double pi; //PI հաստատունը
    double radius; //շառավիղը
```

```
};

//սկզբնարժեքավորում ենք PI հաստատունը
const double Circle::pi=acos(-1.0);

Triangle::Triangle(double a,double b,double c)
    :first(a>=0?a:-a),second(b>=0?b:-b),
     third(c>=0?c:-c)
{
    /*եթե չկա տրված կողմերով եռանկյուն, ապա ուղղել
    ամենամեծ կողմի երկարությունը*/
    if(a+b<c)
        c=a+b;
    if(a+c<b)
        b=a+c;
    if(b+c<a)
        a=b+c;
}

double Triangle::Area() const
{
    double p=(first+second+third)/2;
    return sqrt(p*(p-first)*(p-second)*
                                        (p-third));
}

int main()
{
    const int n=6;
    //ստեղծում ենք պատկերների հասցեների զանգված
    Shape* shapes[n]={&Circle(2),
                      &Triangle(1.2,1.1,2),
                      &Rectangle(3.2,0.5),
                      &Circle(1.5),
```

```
                    &Triangle(2,10,10.1),
                    &Rectangle(0.1,200)};
    //արտածում ենք պատկերների պարագծերը
    for(int i=0;i<n;++i)
    {
        cout<<shapes[i]->Perimeter()<<' ';
    }
    cout<<endl;
    //հիշում ենք մակերեսները այս զանգվածում
    double areas[n];
    for(int i=0;i<n;++i)
    {
        areas[i]=shapes[i]->Area();
    }
    /*պղպջակների ալգորիթմով դասավորում ենք
    պատկերները՝մակերեսների չնվազման կարգով*/
    for(int i=0;i<n-1;++i)
    {
        for(int j=0;j<n-i-1;++j)
            if(areas[j]>areas[j+1])
            {
                swap(areas[j],areas[j+1]);
                swap(shapes[j],shapes[j+1]);
            }
    }
    //արտածում ենք պատկերները
    for(int i=0;i<n;++i)
    {
        shapes[i]->Print();
        cout<<' ';
    }
    cout<<endl;
    return 0;
}
```

Ստորև ներկայացված խնդիրներն առաջարկվում է լուծել ինքնուրույն:

1. Ի՞նչ քերականական սխալներ կան ծրագրի հետևյալ հատվածում.

ա)
```
class Base{
public:
    virtual void f()=0;
};
class A:public Base{
public:
    void func(){ A t; }
    A& h() const { return *this; }
private:
    int b;
};
```

բ)
```
class B{
private:
    B(const B& other){}
};
class A:public B{
public:
    void f(A a) const{ ++b; }
    void g(){ f(*this); }
private:
    int b;
};
```

գ)
```
class Base{
public:
    Base(double d){}
    virtual void h()=0;
};
class A:protected Base{
public:
    A(double d){}
```

```
    void g(){ A a; a.h(); }
};
```

ղ)
```
class A{
protected:
    virtual void h();
private:
    int a;
};
class B:public A{
public:
    B(const A& a){ a.h(); }
    void f(){ a=b; }
private:
    int b;
};
```

2. Ուղղել հետևյալ ծրագրի բոլոր սխալները և նշել, թե ինչ կարտածի այն սխալներն ուղղելուց հետո.

ա)
```
class Time{
    int hr,min;
protected:
    void print()
    { cout<<hr<<':'<<min<<':'; }
public:
    Time(int,int);
};
class TimeS:public Time{
    int sec;
public:
    TimeS(int h,int m,int s)
        :Time(h,m){ sec=s; }
    void print(){
        cout<<hr<<':'<<min
            <<':'<<sec<<endl;
    }
```

```
};
int main(){
    Time t1(20,35); cout<<endl;
    TimeS t2(5,25,40);
    t1.print();
    t2.print();
    return 0;
}
```

բ)
```
class Base{
    int x;
protected:
    void print(){ cout<<x<<endl; }
public:
    Base(int a){ x=a; }
};
class Der:public Base{
    int y;
public:
    Der(int a,int b)
        :Base(a){ y=b; }
    void print(){ cout<<x<<' '<<y<<endl; }
};
int main(){
    Base t1;
    Der t2(10,20);
    t1.print();
    t2.print();
    return 0;
}
```

գ)
```
class B{
    int x;
protected:
    void print(){ cout<<x<<endl; }
public:
    B(int=1);
```

```
    int getx(){ return x; }
};
class D:protected B{
    int y;
public:
    D(int a,int c)
        : B(a){ y=c; }
    void print(){ cout<<x<<' '<<y<<endl; }
};
int main(){
    B b1(15);
    D d1(23,55);
    b1.print();
    d1.print();
    cout<<d1.getx()<<endl;
    return 0;
}
```

3. Իրականացնել հետևյալ դասերը՝ `Point`, `Circle`
   - `Point` դասի օբյեկտները երկչափ տարածության կետեր են,
   - `Circle` դասը ժառանգված է `Point` դասից, և այդ դասի օբյեկտները շրջաններ են (հաստ կետեր):

   Մեթոդներ և գլոբալ ֆունկցիաներ.
   - `Point` դասի կոնստրուկտոր՝ երկու `double` տիպի պարամետրերով, որը ստեղծում է տրված կոորդինատներով կետ (պարամետրերին տալ լռությամբ արժեքներ),
   - `Circle` դասի կոնստրուկտոր՝ երեք `double` տիպի պարամետրերով, որը ստեղծում է կենտրոնի տրված կոորդինատներով և տրված շառավղով շրջան (պարամետրերին տալ լռությամբ արժեքներ),
   - կետը տեղաշարժելու ֆունկցիա,
   - կետի կոորդինատները ստացող ֆունկցիաներ,

- շրջանի շառավղի ստացման և փոփոխման ֆունկցիաներ,
- շրջանի պարագիծը և մակերեսը հաշվող ֆունկցիաներ,
- կետի/շրջանի արտածման ու ներածման ֆունկցիաներ,
- երկու կետի, երկու շրջանի, կետի և շրջանի միջև հեռավորությունը հաշվող ֆունկցիաներ:

Գրել թեստ.

- նկարագրել `Point` և `Circle` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- կառուցել շրջանների զանգված և արտածել դրանք իրենց մակերեսների աճման կարգով,
- կառուցել կետերի ու շրջանների զանգված և արտածել այն կետը, որի նվազագույն հեռավորությունը այդ շրջաններից առավելագույնն է:

4. Իրականացնել հետևյալ դասերը՝ `Person`, `Driver`
   - `Person` դասի օբյեկտները մարդիկ են,
   - `Driver` դասը ժառանգված է `Person` դասից և այդ դասի օբյեկտները վարորդներ են:

   Դաշտեր.

   - `Person` դասում առնվազն պետք է լինեն անուն, ազգանուն, անձնագրի համար և ծննդյան տարեթիվ դաշտերը,
   - `Driver` դասում առնվազն պետք է լինեն վարորդական իրավունքի համարի և տրման տարեթվի դաշտերը:

   Մեթոդներ և գլոբալ ֆունկցիաներ.

   - `Person` և `Driver` դասի օբյեկտներ ստեղծելու համար անհրաժեշտ կոնստրուկտորներ,
   - մարդու անունը, ազգանունը, անձնագրի համարը և ծննդյան տարեթիվը ստացող ու փոփոխող ֆունկցիաներ,
   - վարորդական իրավունքի համարը և տրման տարեթիվը ստացող ու փոփոխող ֆունկցիաներ,

- մարդու տարիքը հաշվող ֆունկցիա,
- վարորդական ստաժը հաշվող ֆունկցիա,
- `Person` և `Driver` դասի օբյեկտների արտածման ու ներածման ֆունկցիաներ:

Գրել թեստ.

- նկարագրել `Person` և `Driver` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- կառուցել վարորդական իրավունքների զանգված և արտածել առավելագույն ստաժ ունեցող վարորդներից նվազագույն տարիքով վարորդի տվյալները:

5. Իրականացնել հետևյալ դասերը՝ `Person`, `Worker`
   - `Person` դասի օբյեկտները մարդիկ են,
   - `Worker` դասը ժառանգված է `Person` դասից, և այդ դասի օբյեկտները աշխատողներ են:

   Դաշտեր.

   - `Person` դասում առնվազն պետք է լինեն անուն, ազգանուն, անձնագրի համար և ծննդյան տարեթիվ դաշտերը,
   - `Worker` դասում առնվազն պետք է լինեն պաշտոն, աշխատավարձ և աշխատանքի ընդունման ամսաթիվ (օր, ամիս, տարի) դաշտերը:

   Մեթոդներ և գլոբալ ֆունկցիաներ.

   - `Person` և `Worker` դասի օբյեկտներ ստեղծելու համար անհրաժեշտ կոնստրուկտորներ,
   - մարդու անունը, ազգանունը, անձնագրի համարը և ծննդյան տարեթիվը ստացող ու փոփոխող ֆունկցիաներ,
   - աշխատողի պաշտոնը, աշխատավարձը և աշխատանքի ընդունման ամսաթիվը (օր, ամիս, տարի) ստացող ու փոփոխող ֆունկցիաներ,

- մարդու տարիքը հաշվող ֆունկցիա,
- աշխատանքային ստաժը հաշվող ֆունկցիա,
- `Person` և `Worker` դասի օբյեկտների արտածման ու ներածման ֆունկցիաներ:

Գրել թեստ.

- նկարագրել `Person` և `Worker` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- կառուցել աշխատողների զանգված և արտածել նրանց միջին աշխատավարձից բարձր աշխատավարձ ստացող աշխատողների տվյալները,
- կառուցել աշխատողների զանգված և արտածել առավելագույն աշխատավարձ ստացող նվազագույն տարիքով աշխատողի տվյալները:

6. Իրականացնել հետևյալ դասերը՝ `Time`, `TimeS`, որոնց օբյեկտներն իրենցից ներկայացնում են ժամանակ:
   - `Time` դասը բազային է, որտեղ ժամանակը տրվում է ժամ (`0`-ից `23`) և րոպե դաշտերով,
   - `TimeS` դասը ժառանգված է `Time` դասից, և դրանում ավելացվում է վայրկյան դաշտը:

   Մեթոդներ և գլոբալ ֆունկցիաներ.

   - կոնստրուկտորներ՝ `int` տիպի պարամետրերով, որը ստեղծում է `Time` և `TimeS` դասի օբյեկտներ,
   - `Time` և `TimeS` տիպի օբյեկտների ներածման ֆունկցիա,
   - `print24` անունով ֆունկցիա, որն արտածում է ժամանակը (`Time` և `TimeS` տիպի օբյեկտները)՝ `24` ժամյա սանդղակով,
   - `print12` անունով ֆունկցիա, որն արտածում է ժամանակը (`Time` և `TimeS` տիպի օբյեկտները)՝ `12` ժամյա սանդղակով,

- `operator==,operator!=,operator>,operator<` (երկու ժամանակների համեմատման ֆունկցիաներ),
- `operator++` (ինկրեմենտի պոստֆիքս և պրեֆիքս տարբերակները), որն ավելացնում է `Time/TimeS` տիպի օբյեկտի արժեքը `1` րոպեյով/վայրկյանով:
- `minuteNumber` անունով ֆունկցիա, որը `Time` տիպի օբյեկտի համար վերադարձնում է, թե օրվա որերորդ րոպեն է այն ներկայացնում,
- `secondNumber` անունով ֆունկցիա, որը `TimeS` տիպի օբյեկտի համար վերադարձնում է, թե օրվա որերորդ վայրկյանն է այն ներկայացնում:

Գրել թեստ.

- նկարագրել `Time, TimeS` տիպի օբյեկտները, կիրառել նշված ֆունկցիաները և արտածել արդյունքները տարբեր տեսքերով,
- ներածել `Time` դասի օբյեկտների զանգված և արտածել կեսօրին ամենամոտ `2` ժամանակները,
- ներածել `TimeS` դասի օբյեկտների զանգված և արտածել ժամանակի ամենաուշ պահը:

7. Իրականացնել հետևյալ դասերը՝ `Book, FictionBook, ScientificBook`
   - `Book` դասի օբյեկտները գրքեր են,
   - `FictionBook` դասը ժառանված է `Book` դասից, և այդ դասի օբյեկտները գեղարվեստական գրքեր են,
   - `ScientificBook` դասը ժառանգված է `Book` դասից, և այդ դասի օբյեկտները գիտական գրքեր են:

   Դաշտեր.

   - `Book` դասում պետք է լինեն վերնագիր ու հեղինակ (անուն ազգանունը) դաշտերը՝ իրականացված, որպես

սիմվոլային տիպի միաչափ զանգված (ստատիկ կամ դինամիկ),

- `Book` դասում պետք է լինեն գին, էջերի քանակ և տպաքանակ դաշտերը՝ ամբողջ տիպի,
- `FictionBook` դասում պետք է լինի ժանր դաշտը՝ թվարկումային տիպի (վեպ, պատմվածք, պոեմ և բանաստեղծություն արժեքներով),
- `FictionBook` դասում պետք է լինի վաճառված օրինակների քանակ դաշտը՝ ամբողջ տիպի,
- `ScientificBook` դասում պետք է լինի բնագավառ դաշտը՝ թվարկումային տիպի (ինֆորմատիկա, ֆիզիկա, մաթեմատիկա, քիմիա և կենսաբանություն արժեքներով),
- `ScientificBook` դասում պետք է լինի գրադարաններին տրված օրինակների քանակ դաշտը՝ ամբողջ տիպի:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- `Book, FictionBook, ScientificBook` դասի օբյեկտներ ստեղծելու համար անհրաժեշտ կոնստրուկտորներ,
- ներածման և արտածման ֆունկցիաներ (ժառանգված դասերի վերաբեռնված ֆունկցիաներում առավելագույնս պետք է օգտագործել բազային դասի ֆունկցիաները),
- `popularity` անունով ֆունկցիա, որը հաշվում է գեղարվեստական գրքի ժողովրդականությունը (որպես սահող ստորակետով թիվ) հետևյալ բանաձևով՝ `(s/c)*(s/1000)`, որտեղ `s`-ը գրքի վաճառված օրինակների քանակն է, իսկ `c`-ն գրքի տպաքանակն է:
- `expensive` անունով ֆունկցիան, որը հաշվում է գիտական գրքի թանկարժեքության աստիճանը (որպես սահող ստորակետով թիվ) հետևյալ բանաձևով՝ գին/էջերի քանակ:

Գրել թեստ.

- նկարագրել `Book`, `FictionBook`, `ScientificBook` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել արդյունքները,
- ներածել `FictionBook` դասի օբյեկտների զանգված, և արտածել այն գեղարվեստական գրքերը, որոնց ժողովրդականությունը մեծ է տրված թվից,
- ներածել `ScientificBook` դասի օբյեկտների զանգված և արտածել այն գրքերը, որոնց թանկարժեքության աստիճանը փոքր է տրված թվից:

8. Իրականացնել հետևյալ դասերը՝ `Vehicle`, `Car`, `Truck`, `TractorTrailer`
   - `Vehicle` դասի օբյեկտները ավտոմեքենաներ են,
   - `Car` դասը ժառանգված է `Vehicle` դասից, և այդ դասի օբյեկտները մարդատար ավտոմեքենաներ են,
   - `Truck` դասը ժառանգված է `Vehicle` դասից, և այդ դասի օբյեկտները բեռնատար ավտոմեքենաներ են,
   - `TractorTrailer` դասը ժառանգված է `Truck` դասից, և այդ դասի օբյեկտները կցորդով բեռնատար ավտոմեքենաներ են:

   Դաշտեր.
   - `Vehicle` դասում առնվազն պետք է լինեն արտադրող, մոդել, գին, քաշ, շարժիչի հզորություն, առավելագույն արագություն և արտադրման տարեթիվ դաշտերը,
   - `Car` դասում պետք է լինի նստատեղերի քանակ դաշտը,
   - `Truck` դասում պետք է լինի բեռնատարողություն դաշտը,
   - `TractorTrailer` դասում պետք է լինի կցորդի երկարություն դաշտը:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- `Vehicle`, `Car`, `Truck` և `TractorTrailer` դասի օբյեկտներ ստեղծելու համար անհրաժեշտ կոնստրուկտորներ,
- վերը նշված դաշտերը ստացող ու փոփոխող ֆունկցիաներ,
- `Vehicle`, `Car`, `Truck` և `TractorTrailer` դասի օբյեկտների արտածման ու ներածման ֆունկցիաներ:

Գրել թեստ.

- նկարագրել `Vehicle`, `Car`, `Truck` և `TractorTrailer` տիպի օբյեկտներ, կիրառել նշված ֆունկցիաները և արտածել դրանք,
- կառուցել մարդատար ավտոմեքենաների զանգվածը և արտածել ամենաշատը 5 նստատեղ ունեցող ավտոմեքենաներից շարժիչի առավելագույն հզորություն ունեցողի տվյալները,
- կառուցել ավտոմեքենաների զանգված (`Vehicle*` տիպի զանգված) և արտածել դրանք իրենց գնի աճման կարգով:

9. Իրականացնել հետևյալ դասերը՝ `Equation`, `LinearEquation`, `QuadradicEquation`
   - `Equation` դասը հանդիսանում է բազային աբստրակտ դաս մնացած դասերի համար, և դրանից ժառանգված դասերի օբյեկտներն իրենցից ներկայացնում են մեկ փոփոխականից հավասարումներ,
   - `LinearEquation` դասի օբյեկտները գծային հավասարումներ են,
   - `QuadradicEquation` դասի օբյեկտները քառակուսի հավասարումներ են:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- `Equation` դասից ժառանգված դասերը պետք է պարունակեն տվյալ դասի օբյեկտ ստեղծելու համար անհրաժեշտ կոնստրուկտոր(ներ) կամ մեթոդներ,
- հավասարման արմատների քանակը և արմատները հաշվող ֆունկցիաներ, որոնք `Equation` դասում պետք է սահմանված լինեն որպես մաքուր վիրտուալ ֆունկցիաներ,
- հավասարումն արտածող ֆունկցիա, որը `Equation` դասում պետք է սահմանված լինի որպես մաքուր վիրտուալ ֆունկցիա:

Գրել թեստ.

- կառուցել քառակուսի հավասարումների զանգված և արտածել դրանց բոլորի արմատների գումարը,
- կառուցել հավասարումների զանգված (`Equation*` տիպի զանգված) և արտածել մեծագույն արմատ ունեցող հավասարումը,
- կառուցել հավասարումների զանգված (`Equation*` տիպի զանգված) և արտածել այն հավասարումը, որն ընդհանուր արմատ ունի առավելագույն թվով հավասարումների հետ:

10. Իրականացնել հետևյալ դասերը՝ `Sequence`, `ArithmeticSequence`, `GeometricSequence`
    - `Sequence` դասը հանդիսանում է բազային աբստրակտ դաս մնացած դասերի համար, և դրանից ժառանգված դասերի օբյեկտներն իրենցից ներկայացնում են թվային շարքեր,
    - `ArithmeticSequence` դասի օբյեկտները թվաբանական պրոգրեսիաներ են,

- `GeometricSequence` դասի օբյեկտները երկրաչափական պրոգրեսիաներ են:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- `Sequence` դասից ժառանգված դասերը պետք է պարունակեն տվյալ դասի օբյեկտ ստեղծելու համար անհրաժեշտ կոնստրուկտոր(ներ) կամ մեթոդներ,
- շարքի տրված կարգահամարով անդամը հաշվող ֆունկցիա, որը `Sequence` դասում պետք է սահմանված լինի որպես մաքուր վիրտուալ ֆունկցիա,
- շարքի սկզբից մինչև տրված կարգահամարով անդամների գումարը հաշվող ֆունկցիա, որը `Sequence` դասում պետք է սահմանված լինի որպես մաքուր վիրտուալ ֆունկցիա:

Գրել թեստ.

- կառուցել թվաբանական պրոգրեսիաների զանգված և արտածել այն պրոգրեսիայի կարգահամարը, որի տասներորդ անդամն առավելագույնն է,
- կառուցել շարքերի զանգված (`Sequence*` տիպի զանգված) և արտածել այն շարքի կարգահամարը, որի սկզբից տրված քանակով անդամների գումարը մեծագույնն է,
- կառուցել շարքերի զանգված (`Sequence*` տիպի զանգված) և արտածել այն շարքի կարգահամարը, որի սկզբի տրված քանակով անդամները մնացած շարքերի սկզբնեերի նույն քանակով անդամների հետ ունեն առավելագույն թվով համընկնումներ:

11. Իրականացնել հետևյալ դասերը՝ `Shape3D`, `Parallelepiped`, `Cylinder`, `Cone`, `Sphere`
    - `Shape3D` դասը հանդիսանում է բազային աբստրակտ դաս մնացած դասերի համար, և դրանից ժառանգված դա-

սերի օբյեկտներն իրենցից ներկայացնում են երկրաչափական եռաչափ պատկերներ,
- `Parallelepiped` դասի օբյեկտները զուգահեռանիստեր են,
- `Cylinder` դասի օբյեկտները գլաններ են,
- `Cone` դասի օբյեկտները կոներ են,
- `Sphere` դասի օբյեկտները գնդեր են:

Մեթոդներ և գլոբալ ֆունկցիաներ.
- `Shape3D` դասից ժառանգված բոլոր դասերը պետք է պարունակեն տվյալ դասի օբյեկտ ստեղծելու համար անհրաժեշտ կոնստրուկտոր(ներ) կամ մեթոդներ,
- պատկերի կողմնային մակերևույթի մակերեսը և ծավալը հաշվող ֆունկցիաներ, որոնք `Shape3D` դասում պետք է սահմանվեն որպես մաքուր վիրտուալ ֆունկցիաներ,
- պատկերն արտածող ֆունկցիա, որը `Shape3D` դասում պետք է սահմանված լինի որպես մաքուր վիրտուալ ֆունկցիա:

Գրել թեստ.
- կառուցել գնդերի զանգված և արտածել դրանք՝ իրենց շառավիղների չնվազման կարգով,
- կառուցել եռաչափ պատկերների զանգված (`Shape3D*` տիպի զանգված) և արտածել բոլորի ծավալները,
- կառուցել եռաչափ պատկերների զանգված (`Shape3D*` տիպի զանգված) և արտածել առավելագույն ծավալ ունեցող պատկերների մեջ նվազագույն կողմնային մակերևույթի մակերես ունեցող օբյեկտը:

12. Իրականացնել հետևյալ դասերը՝ `Transformation`, `RowSwap`, `ColumnSwap`, `RowMult`, `ColumnMult`, `RowAdd`, `ColumnAdd`, `Transpose`, `Rotate`

- `Transformation` դասը հանդիսանում է բազային աբստրակտ դաս մնացած դասերի համար, և դրանից ժառանգված դասերի օբյեկտներն իրենցից ներկայացնում են մատրիցի ձևափոխություններ,
- `RowSwap` դասի օբյեկտները մատրիցի երկու տողերի տեղափոխության ձևափոխություններն են,
- `ColSwap` դասի օբյեկտները մատրիցի երկու սյուների տեղափոխության ձևափոխություններն են,
- `RowMult` դասի օբյեկտները մատրիցի տողը հաստատունով բազմապատկելու ձևափոխություններն են,
- `ColMult` դասի օբյեկտները մատրիցի սյունը հաստատունով բազմապատկելու ձևափոխություններն են,
- `RowAdd` դասի օբյեկտները մատրիցի տողը հաստատունով բազմապատկելու և մեկ այլ տողի գումարելու ձևափոխություններն են,
- `ColAdd` դասի օբյեկտները մատրիցի սյունը հաստատունով բազմապատկելու և մեկ այլ սյան գումարելու ձևափոխություններն են,
- `Transpose` դասի օբյեկտը իրենից ներկայացնում է մատրիցի տրանսպոնացման ձևափոխությունը,
- `Rotate` դասի օբյեկտները մատրիցի 90, 180 կամ 270 աստիճանով պտույտները ներկայացնող ձևափոխություններն են:

<u>Մեթոդներ և գլոբալ ֆունկցիաներ</u>.

- `Transformation` դասից ժառանգված բոլոր դասերը պետք է պարունակեն տվյալ դասի օբյեկտ՝ ստեղծելու համար անհրաժեշտ կոնստրուկտոր(ներ) կամ մեթոդներ,
- ձևափոխությունը տրված երկչափ զանգվածի վրա կիրառելու ֆունկցիա (որը `Transformation` դասում պետք է սահմանված լինի որպես մաքուր վիրտուալ ֆունկցիա):

Գրել թեստ.

- կառուցել մատրիցի տողերի տեղափոխության ձևափոխությունների զանգված (`RowSwap` տիպի զանգված), ներածել մատրից և կիրառելով ձևափոխությունները այդ մատրիցի վրա՝ արտածել ստացված մատրիցը,
- կառուցել մատրիցի ձևափոխությունների զանգված (`Transformation*` տիպի զանգված), ներածել մատրիցներ և կիրառելով ձևափոխությունները այդ մատրիցների վրա՝ արտածել ստացված մատրիցները:

13. Իրականացնել հետևյալ դասերը՝ `Function`, `Linear`, `Sum`, `Mult`, `Exp`, `Sin`, `Cos`, `Log`
    - `Func` դասը հանդիսանում է բազային աբստրակտ դաս մնացած դասերի համար, և դրանից ժառանգված դասերի օբյեկտներն իրենցից ներկայացնում են մեկ փոփոխականի ֆունկցիաներ,
    - `Linear` դասի օբյեկտները $a \cdot x+b$ տեսքի գծային ֆունկցիաներն են, որտեղ `a`-ն և `b`-ն իրական թվեր են,
    - `Sum` դասի օբյեկտները `f(x)+g(x)` տեսքի ֆունկցիաներ են, որտեղ `f(x)`-ը և `g(x)`-ը `Func` դասի ժառանգ դասերի օբյեկտներ են,
    - `Mult` դասի օբյեկտները $f(x) \cdot g(x)$ տեսքի ֆունկցիաներ են, որտեղ `f(x)`-ը և `g(x-ը` `Func` դասի ժառանգ դասերի օբյեկտներ են,
    - `Exp` դասի օբյեկտները $f(x)^{g(x)}$ տեսքի ֆունկցիաներ են, որտեղ `f(x)`-ը և `g(x-ը` `Func` դասի ժառանգ դասերի օբյեկտներ են,
    - `Sin` դասի օբյեկտները `sin(f(x))` տեսքի ֆունկցիաներ են, որտեղ `f(x-ը` `Func` դասի ժառանգ դասի օբյեկտ է,
    - `Cos` դասի օբյեկտները `cos(f(x))` տեսքի ֆունկցիաներ են, որտեղ `f(x-ը` `Func` դասի ժառանգ դասի օբյեկտ է,

- `Log` դասի օբյեկտները $\log_{f(x)}^{g(x)}$ տեսքի ֆունկցիաներ են, որտեղ `f(x-ը` և `g(x-ը` `Func` դասի ժառանգ դասերի օբյեկտներ են:

Մեթոդներ և գլոբալ ֆունկցիաներ.

- `Func` դասից ժառանգված բոլոր դասերը պետք է պարունակեն տվյալ դասի օբյեկտներ՝ ստեղծելու համար անհրաժեշտ կոնստրուկտոր(ներ),
- ֆունկցիայի ածանցյալը հաշվող և վերադարձնող ֆունկցիա, որը `Func` դասում պետք է սահմանված լինի որպես մաքուր վիրտուալ ֆունկցիա,
- `operator()` (հնարավորություն է տալիս ստանալ ֆունկցիայի արժեքը տրված կետում) և պետք է վերաբեռնված լինի միայն `Func` դասի համար,
- `operator<<` (ֆունկցիայի արտածման ֆունկցիա), որը պետք է վերաբեռնված լինի միայն `Func` դասի համար,
- `operator+`, `operator-`, `operator*`, `operator/` (երկու ֆունկցիաների գումարը, տարբերությունը, արտադրյալը և հարաբերությունը ստացող ֆունկցիաներ), որոնք պետք է վերաբեռնված լինեն միայն `Func` դասի համար:

Գրել թեստ.

- կառուցել գծային ֆունկցիաների զանգված և արտածել `YES`, եթե դրանց բոլոր գրաֆիկները հատվում են միևնույն կետում, և `NO`՝ հակառակ դեպքում,
- կառուցել ֆունկցիաների զանգված (`Func*` տիպի զանգված) և արտածել այն ֆունկցիան, որի արժեքը մեծագույնն է տրված կետում,
- կառուցել ֆունկցիաների զանգված (`Func*` տիպի զանգված) և արտածել դրանց ածանցյալները:

Գրականություն

1. Ա. Բ. Աղամյան Ա. Ռ. Բրուտյան Վ. Ս. Եղիազարյան Լ. Ա. Սանոյան Գ. Զ. Սարգսյան, Ծրագրավորման խնդիրների ժողովածու, Երևան, ԵՊՀ հրատ., 1991:
2. Դարբինյան Կ. C++ լեզվի ինքնուսուցման համառոտ ձեռնարկ, Երևան, ԵՊՀ հրատ., 2001:
3. Bjarne Stroustrup. The C++ Programming Language, 3d ed. Reading, MA: Addison-Wesley. 1997 (ռուս. Թարգմ. Бьёрн Страуструп. Язык программирования C++, 3-е изд. — СПб.; М.: Невский диалект — Бином, 1999.)
4. Bjarne Stroustrup. The C++ Programming Language, Special edition. Reading, MA: Addison-Wesley. 1997. (ռուս թարգմ Бьёрн Страуструп. Язык программирования C++. Специальное издание. — М.: Бином-Пресс, 2007.)
5. Brian W. Kernighan, Dennis M. Ritchie. The C Programming Language. Prentice-Hall, 1988. (ռուս. թարգմ. Керниган Б., Ритчи Д. Язык программирования Си: Пер. с англ. — М.: Финансы и статистика, 1992.)
6. Paul J. Deitel, Harvey Deitel. C++ How to Program, 10th Edition, Pearson, 2016 (ռուս. թարգմ. Х. М. Дейтел, П. Дж. Дейтел Как программировать на C++, 5-е изд. — М.: Бином, 2008.)
7. Meyers, Scott. 1996. More Effective C++: 35 New Ways to Improve Your Programs and Designs. Reading, MA: Addison-Wesley. (ռուս. թարգմ. Майерс С. Эффективное использование C++. 35 новых способов улучшить стиль программирования. — СПб: Питер, 2006.)
8. Meyers, Scott. 1998. Effective C++: 50 Specific Ways to Improve Your Programs and Designs, 2d ed. Reading, MA: Addison-Wesley. (ռուս. թարգմ. Майерс С. Эффективное использование C++. 50 рекомен-

даций по улучшению ваших программ и проектов. — СПб: Питер, 2006.)
9. Stephen Prata. C Primer Plus, 6th Ed. (Developer's Library). — Pearson, 2011. (рус. пер. Стивен Прата. Язык программирования C++ (C++11). — М.: Вильямс, 2012.)
10. Павловская Т. А. C/C++. Программирование на языке высокого уровня. — СПб: Питер, 2006.
11. Подбельский В. В. Язык C++, 5-е изд. — М. Финансы и статистика, 2003.

ԵՐԵՎԱՆԻ ՊԵՏԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Լ. Օ. Խաչոյան, Ա. Ս. Անդրեասյան
Ա. Հ. Առաքելյան, Վ. Ա. Զաքարյան
Ա. Գ. Մանուկյան, Ջ. Ա. Նազարյան

ԾՐԱԳՐԱՎՈՐՈՒՄ C++ ԼԵԶՎՈՎ

(խնդիրների ժողովածու)

Համակարգչային ձևավորումը՝ Կ. Չալաբյանի
Կազմի ձևավորումը՝ Ա. Պատվականյանի
Հրատ. խմբագրումը՝ Լ. Հովհաննիսյանի

Տպագրված է «Time to Print» օպերատիվ տպագրությունների սրահում:
Ք. Երևան, Խանջյան 15/55

Ստորագրված է տպագրության՝ 17.12.2019:
Չափսը՝ 60x84 1/16: Տպ. մամուլը՝ 14,625:
Տպաքանակը՝ 200:

ԵՊՀ հրատարակչություն
ք. Երևան, 0025, Ալեք Մանուկյան 1
www.publishing.am

ՀՐԱՏԱՐԱԿՉՈՒԹՅՈՒՆ
ԵՐԵՎԱՆ 2019
publishing.ysu.am