Kantonalno takmičenje iz informatike za učenike osnovnih škola KS 2017. godine

Sarajevo, 22. april 2017.

Napomena o načinu bodovanja zadataka i programskom jeziku BASIC

Nastavni planovi i programi informatike predviđaju upotrebu programskog jezika BASIC, a u većini škola se koristi okruženje QBasic za DOS ili noviji QB64. No u posljednjih 10ak godina na državnim takmičenjima, a od 2014. godine i na Kantonalnom takmičenju, zadaci se pregledaju automatski. Ovo nije moguće realizirati koristeći QBasic ili QB64, pa se za te potrebe koristi FreeBasic u "QBasic modu".

Iako autori softvera FreeBasic garantuju da je uz korištenje "QBasic moda" ovaj program 99% kompatibilan sa QBasic-om, postoje neke sitne razlike na koje želimo da skrenemo pažnju:

- **U FreeBasic-u nije moguće deklarisati polje (niz) i promjenljivu (varijablu) koji se isto zovu**

  U QBasic-u ako imate sljedeći kod:

  ```
  DIM a$(50)

  INPUT a$
  ```

  on će ispravno raditi budući da su polje (niz) a$ i string a$ različite varijable, a stringovi se ne moraju deklarisati. Nažalost, u programima često možemo vidjeti da učenik misli da linijom DIM a$(50) deklariše string dužine 50 znakova. Linija DIM a$(50) ustvari deklariše niz od 50 stringova! FreeBasic neće dozvoliti postojanje niza i varijable sa istim imenom, pa će kod dat iznad na serveru proizvesti Compiler error.

- S druge strane, **nizovi (polja) se moraju dimenzionisati.**

  Primjećujemo da neki dijalekti QBasica dozvoljavaju korištenje nizova bez dimenzionisanja u nekim slučajevima. U FreeBasicu se svi korišteni nizovi moraju propisno deklarisati naredbom DIM.

- **Ograničenja broja karaktera na ulazu**

  Kao što je poznato u QBasic-u se ne može u string unijeti više od oko 200 karaktera. FreeBasic dozvoljava unos do 4096 karaktera. Da biste izbjegli ove probleme, ako se u zadatku traži unos velikog broja karaktera, prepravite vaš program tako da umjesto sa standardnog ulaza čita podatke iz datoteke **input.txt**.

- Posebno molimo takmičare da ne koriste naredbu LINE INPUT za čitanje sa standardnog ulaza zbog poznatog buga u FreeBasic okruženju. Molimo vas da za standardni ulaz koristite ili naredbu INPUT ili datoteku **input.txt**.

Cifre

U raznim oblastima računarstva (npr. kriptografija) koriste se brojevi sa vrlo velikim brojem cifara, znatno većim nego što može stati u tipove promjenljivih koje podržavaju programski jezici koje koristimo. Jedan način kako možemo unijeti takav broj je da unesemo niz prirodnih brojeva koje treba posmatrati kao jedan neprekinut broj. Npr. korisnik je unio sljedeće brojeve:

```
233 38 17777 7787 152 -1
```

Broj -1 označava kraj unosa, a ostali brojevi su pozitivni (što nije potrebno provjeravati). Radi uštede prostora brojeve smo razdvojili razmakom, ali ustvari korisnik će nakon svakog unesenog broja pritisnuti tipku Enter. Ovaj niz brojeva ustvari označava sljedeći veliki broj:

```
23338177777787152
```

Ovakav broj može zauzimati i dosta memorije. Pošto u nizu ima dosta ponavljanja, neko se dosjetio da se veliki broj može kraće zapisati ako umjesto niza cifara ispišemo cifru a zatim broj koji označava koliko puta se ona uzastopno ponavlja u nizu. Na primjer, broj koji smo naveli iznad mogao bi se predstaviti ovako:

```
2 1 3 3 8 1 1 1 7 6 8 1 7 1 1 1 5 1 2 1
```

Cifra 2 se ponavlja jednom uzastopno (obratite pažnju da imamo još jednu dvicu na kraju, ali ovdje nas interesuje samo broj **uzastopnih** ponavljanja na trenutnom mjestu u nizu), zatim cifra 3 se ponavlja 3 puta, zatim cifra 8 jednom, cifra 1 jednom, cifra 7 šest puta itd.

Napišite program koji omogućuje unos niza prirodnih brojeva kao u prvom primjeru. Niz neće imati više od 1000 elemenata, a unos se završava brojem -1. Nije potrebno provjeravati da li je uneseno više od 1000 brojeva ili da li su uneseni brojevi manji od -1. Nakon toga program treba ispisati niz parova "cifra – broj uzastopnih ponavljanja", kao u posljednjem primjeru.

Napominjemo da se ovaj zadatak može riješiti bez posebnih struktura za predstavljanje brojeva sa vrlo velikim brojem cifara (bigint)! *Sve što treba je prebrojavati cifre uzastopno!*

Format ulaza

Na ulazu se nalazi određeni broj prirodnih brojeva (ne više od 1000) koji se završava brojem -1. Brojevi su razdvojeni oznakom za novi red (tipka Enter).

Format izlaza

Na izlazu se nalaze parovi prirodnih brojeva od kojih prvi predstavlja cifru a drugi broj uzastopnih ponavljanja te cifre. Brojevi trebaju biti međusobno razdvojeni znakom razmaka.

Ograničenja

U nizu se može nalaziti najviše hiljadu brojeva. Program se ne smije izvršavati duže od 10s niti koristiti više od 1MB memorije.

Primjer

Ulaz:	*Izlaz:*
233 38 17777 7787 152 -1	2 1 3 3 8 1 1 1 7 6 8 1 7 1 1 1 5 1 2 1

U primjeru izlaza došlo je do prelaska u sljedeći red zbog širine predviđenog prostora, no ovdje sve treba biti ispisano u istom redu.

Rješenje (C++):

```
#include <iostream>

using namespace std;

int main() {
	int brojevi[1000];
	int vel=0;

	// Unos brojeva
	for (int i=0; i<1000; i++) {
		cin >> brojevi[i];
		if (brojevi[i] == -1) break;
		vel++;
	}

	// Prolazimo kroz niz
	int prethodna = -1, broj=0;
	for (int i=0; i<vel; i++) {
		// Okrecemo broj naopako
		int naopako=0, nule=0;
		do {
			int cifra=brojevi[i]%10;
			naopako = naopako*10 + cifra;
			if (naopako==0 && cifra==0) nule++;
			brojevi[i] = brojevi[i] / 10;
		} while (brojevi[i] > 0);

		// Prolazimo kroz cifre, uzimajuci i nulu kao cifru
		while (naopako>0) {
			int cifra = naopako%10;
```

```
            if (cifra == prethodna) {
                broj++;
            } else {
                if (prethodna != -1)  // Pocetnu -1 ne ispisujemo
                    cout << prethodna << " " << broj << " ";
                prethodna = cifra;
                broj = 1;
            }
            naopako = naopako / 10;
        }

        // Specijalni slucaj za pocetne nule
        if (nule > 0) {
            if (prethodna == 0) broj += nule;
            else {
                if (prethodna != -1)  // Pocetnu -1 ne ispisujemo
                    cout << prethodna << " " << broj << " ";
                prethodna = 0;
                broj = nule;
            }
        }
    }
    // Ispisujemo i posljednju cifru
    cout << prethodna << " " << broj;

    return 0;
}
```

Rješenje (QBasic):

```
DIM brojevi(1000)
vel = 0
FOR i = 1 TO 1000
    INPUT brojevi(i)
    IF brojevi(i) = -1 THEN
        EXIT FOR
    END IF
    vel = vel + 1
NEXT i

prethodna = -1
broj = 0
FOR i = 1 TO vel
    naopako = 0
    nule = 0
    DO
        cifra = brojevi(i) MOD 10
        naopako = naopako * 10 + cifra
        IF naopako = 0 AND cifra = 0 THEN
            nule = nule + 1
        END IF
```

```
        brojevi(i) = brojevi(i) \ 10
    LOOP WHILE brojevi(i) >= 1

    WHILE naopako >= 1
        cifra = naopako MOD 10
        IF cifra = prethodna THEN
            broj = broj + 1
        ELSE
            IF prethodna > -1 THEN
                PRINT prethodna; broj;
            END IF
            prethodna = cifra
            broj = 1
        END IF
        naopako = naopako \ 10
    WEND

    IF nule > 0 THEN
        IF prethodna = 0 THEN
            broj = broj + nule
        ELSE
            IF prethodna > -1 THEN
                PRINT prethodna; broj;
            END IF
            prethodna = 0
            broj = nule
        END IF
    END IF
NEXT i

PRINT prethodna; broj
```

Testni primjeri

Ulaz:	*Izlaz:*
Primjer 1: 1 2 3 4 5 6 7 8 9 -1	**Primjer 1:** 1 1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9 1
Primjer 2: 1234 5678 9 -1	**Primjer 2:** 1 1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9 1

Primjer 3: 111111 -1	**Primjer 3:** 1 6
Primjer 4: 3333 3333 4444 44444 -1	**Primjer 4:** 3 8 4 9
Primjer 5: 1234 4444 4567 -1	**Primjer 5:** 1 1 2 1 3 1 4 6 5 1 6 1 7 1
Primjer 6: 8765 543 12 -1	**Primjer 6:** 8 1 7 1 6 1 5 2 4 1 3 1 1 1 2 1
Primjer 7: 6767 8888 8765 -1	**Primjer 7:** 6 1 7 1 6 1 7 1 8 5 7 1 6 1 5 1
Primjer 8: 1230 1000 300 0 1230 -1	**Primjer 8:** 1 1 2 1 3 1 0 1 1 1 0 3 3 1 0 3 1 1 2 1 3 1 0 1
Primjeri 9 i 10 nisu navedeni zbog manjka prostora	

Ista slova

Učenici VII_1 razreda žele se igrati neke igre u kojoj učestvuju parovi takmičara, ali nisu se mogli dogovoriti kako da se podijele u parove. Zbog toga su smislili sljedeći način: par mogu činiti dvoje učenika čija se imena sastoje od istih slova.

Npr. imena ELMA i AMELA se sastoje od istih slova: A, E, L, M. Broj koliko puta se koje slovo ponavlja nije bitan – bitno je da se sva slova koja se nalaze u jednom imenu nalaze i u drugom imenu, i obrnuto. Npr. ANES i SENAD ne čine par jer se u imenu SENAD nalazi slovo D kojeg nema u imenu ANES, ali oni ne bi bili par ni kada bismo im zamijenili mjesta tako da je prvi član para SENAD a drugi ANES.

Napravite program koji omogućuje da se unese niz parova imena i provjerava da li svaki od tih parova može činiti tim ili ne može. Prilikom unosa mogu se koristiti i velika i mala slova, a program treba zanemariti razliku između njih.

U prvom redu ulaza nalazi se prirodan broj N (nije potrebno provjeravati da li je prirodan) koji označava koliko parova će biti uneseno. U narednih 2N redova nalaze se imena. Imena će se sastojati isključivo od velikih i malih slova. Za svaka dva imena treba ispisati DA ako ta dva imena čine par ili NE ako ne čine.

Format ulaza

Na ulazu se najprije nalazi prirodan broj N koji predstavlja broj parova. U narednih 2N redova nalaze se imena koja se sastoje isključivo od velikih i malih slova, ne duža od 20 slova.

Format izlaza

Potrebno je u N redova ispisati tekst DA ili NE ovisno o tome da li odgovarajući par može činiti tim ili ne može. Svejedno je da li se tekst DA ili NE ispisuje odmah nakon unosa para ili tek nakon unosa svih parova.

Ograničenja

Biće uneseno najviše 1000 parova. Program se ne smije izvršavati duže od 10s niti koristiti više od 1MB memorije.

Primjer

Ulaz:	*Izlaz:*
2 Amela ELMA Senad Anes	DA NE

Rješenje (C++):

```
#include <iostream>
#include <string>

using namespace std;

int main() {
	int broj_parova;
	string ime1, ime2;
	cin >> broj_parova;
	bool ima_slovo1[30], ima_slovo2[30];

	for (int i=0; i<broj_parova; i++) {
		cin >> ime1;
		cin >> ime2;

		// Postavljamo status za sva slova na false
		for (int j=0; j<30; j++) {
			ima_slovo1[j] = false;
			ima_slovo2[j] = false;
		}

		// Prolazimo kroz prvi string
		for (int j=0; j<ime1.length(); j++) {
			// Promjenljiva idx je indeks u nizu statusa
			int idx;
			// Ako je veliko slovo, indeks dobijamo oduzimanjem velikog slova A
			if (ime1[j] >= 'A' && ime1[j] <= 'Z') idx = ime1[j] - 'A';
			// Ako je malo, oduzimamo malo slovo a
			if (ime1[j] >= 'a' && ime1[j] <= 'z') idx = ime1[j] - 'a';

			// Postavljamo status na ima
			ima_slovo1[idx] = true;
		}

		// Isto to za drugi string
		for (int j=0; j<ime2.length(); j++) {
			int idx;
			if (ime2[j] >= 'A' && ime2[j] <= 'Z') idx = ime2[j] - 'A';
			if (ime2[j] >= 'a' && ime2[j] <= 'z') idx = ime2[j] - 'a';
			ima_slovo2[idx] = true;
		}

		// Dva imena se sastoje od istih slova ako su statusi za sva slova isti
		bool ista_slova=true;
		for (int j=0; j<30; j++) {
			if (ima_slovo1[j] != ima_slovo2[j])
				ista_slova=false;
		}
		if (ista_slova) cout << "DA" << endl; else cout << "NE" << endl;
```

```
    }
    return 0;
}
```

Rješenje (QBasic):

```
CLS
INPUT n
FOR i = 1 TO n
    DIM slovaa(30)
    DIM slovab(30)
    FOR j = 1 TO 30
        slovaa(j) = 0
        slovab(j) = 0
    NEXT j

    INPUT a$
    INPUT b$
    FOR j = 1 TO LEN(a$)
        slovo$ = MID$(a$, j, 1)
        IF ASC(slovo$) > 64 AND ASC(slovo$) < 91 THEN
            slovaa(ASC(slovo$) - 64) = 1
        ELSEIF ASC(slovo$) > 96 AND ASC(slovo$) < 123 THEN
            slovaa(ASC(slovo$) - 96) = 1
        END IF
    NEXT j
    FOR j = 1 TO LEN(b$)
        slovo$ = MID$(b$, j, 1)
        IF ASC(slovo$) > 64 AND ASC(slovo$) < 91 THEN
            slovab(ASC(slovo$) - 64) = 1
        ELSEIF ASC(slovo$) > 96 AND ASC(slovo$) < 123 THEN
            slovab(ASC(slovo$) - 96) = 1
        END IF
    NEXT j

    jednaka = 1
    FOR j = 1 TO 30
        IF (slovaa(j) <> slovab(j)) THEN
            jednaka = 0
        END IF
    NEXT j
    IF jednaka = 1 THEN
        PRINT "DA"
    ELSE
        PRINT "NE"
    END IF

NEXT i
```

Testni primjeri

Ulaz:	*Izlaz:*
Primjer 1: 10 A A B B C C D D E E F F G G H H I I J A	**Primjer 1:** DA DA DA DA DA DA DA DA DA NE
Primjer 2: 4 A AAAA AAAAAAAAAAAA AA Abbccc AAAbcc X Y	**Primjer 2:** DA DA DA NE
Primjer 3: 4 a A Bbc bCc ZvUt zVuT eeeeeee f	**Primjer 3:** DA DA DA NE
Primjer 4: 3 abcde cdba qwertyuiopasdfghjk kjhgfdsapoiuytrewq zvut utzv	**Primjer 4:** NE DA DA
Primjer 5: 4	**Primjer 5:** NE

qwertyuiop	NE
qwertyuiopa	NE
asdfghjkl	DA
asdghjkl	
zxcvbnm	
xcvbnm	
qazwsxedc	
qazwsxecd	
Primjer 6:	**Primjer 6:**
4	DA
AaaAbBBccC	DA
abc	NE
abc	NE
AaaAbBBccC	
qQQqwEeeerRRr	
rrRreEeeeqqQ	
rrRreEeeeqqQ	
qQQqwEeeerRRr	
Primjer 7:	**Primjer 7:**
4	NE
iajdsfacnwuralijdfa	NE
opriwpasdjfaoiwermx	DA
iiqwefpnaoisjornaer	DA
asmerpaowieerunaopi	
tqewrtrqwreqrewrerr	
tqwretrqwerrqwetrwe	
kgjhhjkgjhkjhgkjkjg	
hgjkgjhkjgkhjkjkgjk	
Primjer 8:	**Primjer 8:**
5	NE
abcdefgh	NE
bcdefgh	NE
bcdefgh	NE
abcdefgh	DA
yxzrstuvw	
tuvwyxrs	
tuvwyxrs	
yxzrstuvw	
A	
A	
Primjeri 9 i 10 nisu navedeni zbog manjka prostora	

Bilijar

Svako kome je poznata igra bilijar zna da se kugla kreće po stolu i odbija od rubove stola slijedeći jednostavno pravilo: ugao pod kojim će se kugla odbiti od rub jednak je uglu pod kojim je kugla prilazila rubu.

U našem zadatku zamislićemo da je dat neki bilijarski sto dimenzija $n \cdot m$ (n je širina, a m dužina stola). Neka su na stolu iscrtane linije tako da on izgleda kao koordinatni sistem. Na početku igrač postavlja kuglu na mjesto tako što izbroji x_1 linija po širini i y_1 linija po dužini. Zatim cilja bilijarskim štapom tako da pogodi ivicu stola na mjestu x_P i y_P.

Pretpostavimo da će se kugla kretati beskonačno dugo i odbijati od rubove po pravilu koje smo spomenuli. Koliko puta će se kugla odbiti od ivice stola prije nego što se po prvi put vrati u početnu poziciju?

Ukoliko kugla pogodi ćošak stola, tada problem nema rješenja i program treba ispisati -1 M gdje je M broj odbijanja od ivice prije pogađanja ćoška, u suprotnom program treba ispisati 0 M gdje je M broj odbijanja prije povratka u početnu poziciju.

Format ulaza

U prvom redu ulaza se nalaze četiri broja: x_1 y_1 x_P i y_P koji predstavljaju redom koordinate početne pozicije kugle i koordinate tačke ivice stola koju igrač cilja da pogodi. U sljedećem redu se nalaze dva cijela broja n i m razdvojena razmakom, koji predstavljaju širinu i dužinu stola.

Format izlaza

Na izlazu treba ispisati dva cijela broja **b** i **M** razdvojena razmakom. Broj **b** ima vrijednost -1 ako je kugla pogodila ćošak stola, a **M** broj odbijanja prije pogađanja ćoška. Ako kugla nije pogodila ćošak broj **b** ima vrijednost 0 a broj **M** je broj odbijanja prije povratka u početnu poziciju.

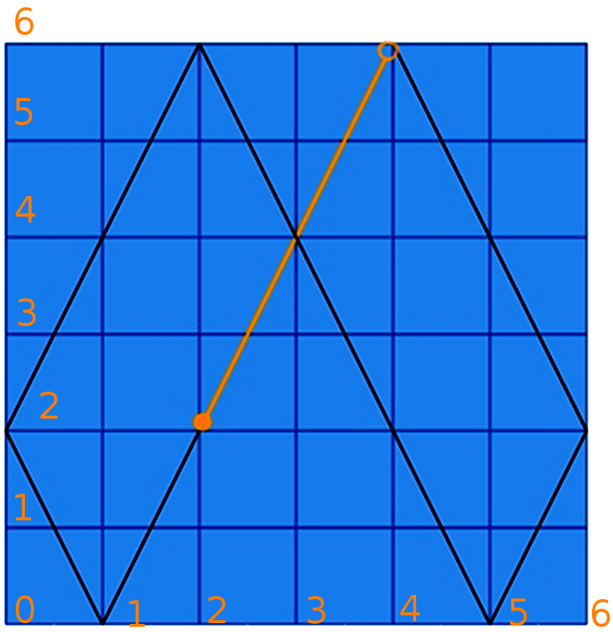
Ograničenja

Program se ne smije izvršavati duže od 10s niti koristiti više od 1MB memorije.

Primjer

Ulaz:	*Izlaz:*
2 2 4 6 6 6	0 6

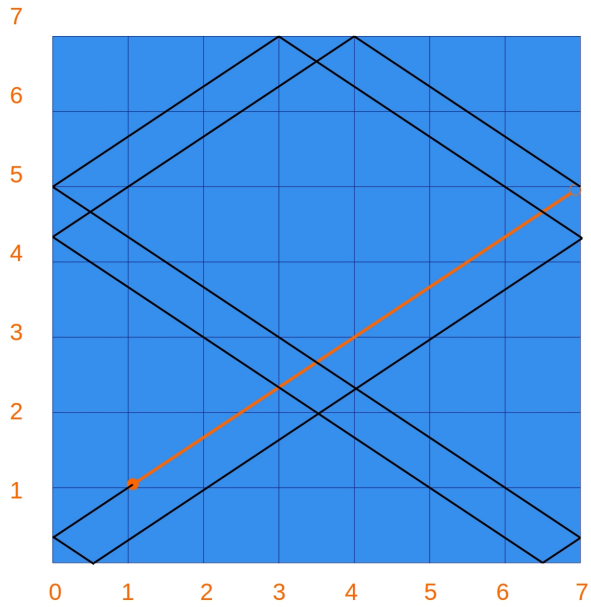
U ovom primjeru kugla se nalazi na početnoj poziciji 2,2 (obojena narandžasta tačka na slici). Igrač cilja tačku na rubu stola 4,6 (narandžasta kružnica na slici). Ukupne dimenzije stola su 6x6.



Na izlazu trebamo dobiti 0 6. Broj 0 označava da postoji rješenje, a broj 6 da se kugla 6 puta odbila od rubove stola. Kugla se odbija od rubove redom u tačkama: (4,6), (6,2), (5,0), (2,6), (0,2), (1,0). Iako se kugla uvijek na početku nalazi na cjelobrojnim koordinatama i cilja se prema tački koja je također na cjelobrojnim koordinatama, to ne znači da će se sve buduće tačke odbijanja nalaziti na cjelobrojnim koordinatama.

Primjer 2:

Ulaz:	*Izlaz:*
1 1 7 5 7 7	0 10



Rješenje (C++):

```
#include <iostream>

using namespace std;

long long int x1,y1, xp,yp, n, m,K1(0),K2(0);

unsigned int abs(long long int x){
   return (x>0)?x:-x;
}
long double absDbl(long double dbl){
   return (dbl>0)?dbl:-dbl;
}
long long int dajTren(long long int K, long long int M, long long int M0){
   long long int rez;
   if(K==0)rez = M0;
   else
   if(K==-1)rez = -M0;
   else{
      if(K%2){
         rez=(K+1)*M-M0;
      }else{
         rez=K*M+M0;
      }
   }
   return rez;
}

void traziK1iK2(){
      //K1 broj odbijanja od lijevu/desnu ivicu stola
      //K2 broj odbijanja od gornju/donju ivicu stola

      //sl je koeficient pravca prave koja prolazi kroz (x1,y1) i (xp,yp)
      static long double sl=(double)(yp-y1)/(double)(xp-x1);

      static bool pravacX = (xp<x1);// false kugla ide prema lijevoj ivici, true prema desnoj
      static bool pravacY = (yp<y1);// false kugla ide prema donjoj ivici, true prema gornjoj

      bool izadji=false;
      while(!izadji){
         long long int cosakX,cosakY; // trenutni najbliži čošak u pracu kretanja kugle
         long long int probnix(dajTren(K1,n,x1)),probniy(dajTren(K2,m,y1)); //trenutna pozicija rješenja
         long double desno = (long double)(probnix-x1)*sl;   //desna strana jednačine kojom se
provjerava
                                          //  da li je nađeno rješenje
         long double lijevo = (long double)(probniy-y1);     //lijeva strana jednačine

         if((K1!=0||K2!=0)&&absDbl(lijevo-desno)<0.0000000001){
            std::cout<<0<<" "<<abs(K1)+abs(K2)<<"\n";
            izadji=true;continue;//pronadjeno rješenje
```

```
        }else{
          cosakX=(K1+1-pravacX)*n;
          cosakY=(K2+1-pravacY)*m;

          if((long double)cosakY>((long double)(cosakX-x1)*sl+(long double)(y1))){//pomjera se gore
            if(!pravacY)K1+=(pravacX)?(-1):1;   // K1 se povecava ukoliko se kugla odbija od desnu ivicu
                                                // K1 se smanjuje ukoliko se kugla odbija od lijevu ivicu
            else K2--;//K2 se smanjuje jer se kugla odbija od donju ivicu
          }else if((long double)cosakY<((long double)(cosakX-x1)*sl+(long double)(y1))){
            if(!pravacY)K2++;          // K2 se povecava jer se kugla odbija od gornju ivicu
            else K1+=(pravacX)?(-1):1;
          }else{
            std::cout<<-1<<" "<<(abs(K1)+abs(K2))<<"\n";
            izadji=true;continue; //pogodjeno je u cosak pa se pretraga prekida
          }
        }
      }
}
int main()
{
   std::cin>>x1>>y1>>xp>>yp>>n>>m;
   if(x1==xp||y1==yp) cout<<0<<" "<<1;
   else traziK1iK2();
   return 0;
}
```

Ispod navodimo i varijantu rješenja u kojoj se koristi rekurzija.

```
#include <iostream>

using namespace std;

long long int x1,y1, xp,yp, n, m,K1(0),K2(0);

unsigned int my_abs(long long int x){
   return (x>0)?x:-x;
}
long double absDbl(long double dbl){
   return (dbl>0)?dbl:-dbl;
}
long long int dajTren(long long int K, long long int M, long long int M0){
   long long int rez;
   if(K==0)rez = M0;
   else
   if(K==-1)rez = -M0;
   else{
      if(K%2){
         rez=(K+1)*M-M0;
      }else{
         rez=K*M+M0;
      }
   }
```

```
    return rez;
}
int traziK1iK2(){
      static long double sl=(double)(yp-y1)/(double)(xp-x1);

      static bool pravacX = (xp<x1);// false negativan smjer true pozitivan
      static bool pravacY = (yp<y1);// false negativan smjer true pozitivan
      long long int cosakX,cosakY;
      long long int probnix(dajTren(K1,n,x1)),probniy(dajTren(K2,m,y1));
      cout.precision(17);

      long double desno = (long double)(probnix-x1)*sl;
      long double lijevo = (long double)(probniy-y1);
      if((K1!=0||K2!=0)&&absDbl(lijevo-desno)<0.0000000001){
         std::cout<<0<<" ";
         return my_abs(K1)+my_abs(K2);
      }else{
      cosakX=(K1+1-pravacX)*n;
      cosakY=(K2+1-pravacY)*m;

      if((long double)cosakY>((long double)(cosakX-x1)*sl+(long double)(y1))){//pomjera se gore
         if(!pravacY)K1+=(pravacX)?(-1):1;
         else K2--;
      }else if((long double)cosakY<((long double)(cosakX-x1)*sl+(long double)(y1))){
         if(!pravacY)K2++;
         else K1+=(pravacX)?(-1):1;
      }else{
         std::cout<<-1<<" ";
         return my_abs(K1)+my_abs(K2); //pogodili u cosak pa se pretraga prekida
      }
      return traziK1iK2();
      }
}
int main()
{
   std::cin>>x1>>y1>>xp>>yp>>n>>m;
   if(x1==xp||y1==yp) cout<<0<<" "<<1;
   else std::cout<<traziK1iK2();
   return 0;
}
```

Testni primjeri

Ulaz:	*Izlaz:*

Primjer 1:
`1 1 2 30000`
`30000 30000`
	Primjer 1:
	`0 60000`

Primjer 2:
`2 2 1 0`
`6 6`
	Primjer 2:
	`0 6`

Primjer 3: `1 1 2 2`	**Primjer 3:** `-1 0`

2 2 **Primjer 4:** 1 1 1 6 6 6 **Primjer 5:** 2 2 0 4 6 6 **Primjer 6:** 1 1 3 13 13 13 **Primjer 7:** 1 1 5 7 7 7 **Primjer 8:** 2 2 3 111 111 111 **Primjer 9:** 4 4 5 227 277 227 **Primjer 10:** 1 1 2 2 100000 2	**Primjer 4:** 0 1 **Primjer 5:** 0 4 **Primjer 6:** 0 14 **Primjer 7:** 0 10 **Primjer 8:** 0 220 **Primjer 9:** -1 43420 **Primjer 10:** -1 49999

V-niz

Pod pojmom v-niz smatramo niz brojeva koji je strogo opadajući (svaki član niza je manji i nejednak prethodnom članu niza) sve do nekog minimalnog člana niza, a od tog minimalnog člana do kraja je strogo rastući (svaki član niza je veći i nejednak prethodnom članu niza). Npr. sljedeći niz brojeva je v-niz:

5, 3, 1, 2, 4

Poseban slučaj čini niz koji je kompletan strogo opadajući (posljednji član je najmanji), i niz koji je kompletan strogo rastući (prvi član je najmanji). Ova dva niza logično nisu v-nizovi.

Napišite program koji omogućuje korisniku da unese jedan niz realnih brojeva. Najprije se unosi prirodan broj N koji predstavlja broj članova niza (ne veći od 100), a zatim se unosi N realnih brojeva. Program treba ispisati poruku NE ako niz nije v-niz, a ako jeste v-niz treba ispisati redni broj člana niza koji je najmanji u tom nizu. Npr. za primjer iznad treba ispisati broj 3 jer je najmanji član treći po redu.

Program koji u svim slučajevima ispisuje poruku NE biće bodovan sa 0 bodova bez obzira što ispunjava određeni broj testova!

Format ulaza

Na ulazu se najprije nalazi prirodan broj N koji predstavlja broj članova niza. U svakom sljedećem redu nalazi se po jedan realan broj koji predstavlja član niza.

Format izlaza

Potrebno je ispisati tekst NE ako niz nije v-niz, ili ako jeste redni broj člana niza koji je najmanji (pri čemu prvi član niza ima redni broj 1).

Ograničenja

Niz će imati najviše 100 elemenata. Program se ne smije izvršavati duže od 10s niti koristiti više od 1MB memorije.

Primjer 1:

Ulaz:	*Izlaz:*
5 5.1 3.1 1.1 2.1 4.1	3

Primjer 2:

Ulaz:	*Izlaz:*
3 10 10 10	NE

Rješenje (C++):

```
#include <iostream>

using namespace std;

int main() {
	double niz[100];
	int vel;

	cin >> vel;

	// Unosimo niz
	for (int i=0; i<vel; i++) {
		cin >> niz[i];
	}

	// Trazimo minimum
	int min=-1;
	// Ako je prvi clan niza veci od drugog, odmah znamo da ovo nije v-niz
	if (niz[1] > niz[0]) {
		cout << "NE";
		return 0;
	}
	for (int i=1; i<vel; i++) {
		// Ako su dva susjedna clana jednaka, ovo nije v-niz
		if (niz[i] == niz[i-1]) { cout << "NE"; return 0; }
		// Ako je clan veci od prethodnog i do sada nismo nasli minimum, prethodni je minimum
		if (niz[i] > niz[i-1]) {
			if (min==-1) min=i-1;
		}
		// Ako je niz jednom rastao vise ne smije ponovo opadati
		if (niz[i] < niz[i-1] && min > -1) {
			cout << "NE"; return 0;
		}
	}
	if (min == -1)
		cout << "NE";
	else
		cout << min+1;
```

```
    return 0;
}
```

Rješenje (QBasic):

```
CLS
DIM niz(100)
INPUT n
FOR i = 1 TO n
    INPUT niz(i)
NEXT i

min=0
IF niz(2)>niz(1) THEN
    PRINT "NE"
    END
END IF
FOR i = 2 TO n
    IF niz(i) = niz(i-1) THEN
        PRINT "NE"
        END
    END IF
    IF niz(i) > niz(i-1) AND min = 0 THEN
        min = i - 1
    END IF
    IF niz(i) < niz(i-1) AND min > 0 THEN
        PRINT "NE"
        END
    END IF
NEXT i
IF min=0 THEN
    PRINT "NE"
ELSE
    PRINT i
END IF
```

Testni primjeri

Ulaz:	*Izlaz:*
Primjer 1: 3 0.1 0.1 0.1	**Primjer 1:** NE
Primjer 2: 4 6 5 4 3	**Primjer 2:** NE
Primjer 3:	**Primjer 3:**

5 1.1 1.2 1.3 1.4 1.5	NE
Primjer 4: 4 4 3 4 3	**Primjer 4:** NE
Primjer 5: 4 1 2 1 2	**Primjer 5:** NE
Primjer 6: 7 5 4 3 4 3 4 5	**Primjer 6:** NE
Primjer 7: 7 0.4 0 -2 -4 -3 1 100	**Primjer 7:** 4
Primjer 8 nije naveden zbog manjka prostora	
Primjer 9: 4 0.0003 0.0002 0.0001 0.0002	**Primjer 9:** 3
Primjer 10: 6 -4 -5 -7 -6 -2 -1	**Primjer 10:** 3