Školsko natjecanje iz informatike

Srednja škola

Prva podskupina (1. i 2. razred)

25. siječnja 2019.

ime zadatka	**OTOPINA**	**STUBE**	**FIGURA**
vremensko ograničenje	5 sekundi	5 sekundi	5 sekundi
broj bodova	40	50	60
	150		

Ministarstvo znanosti i obrazovanja

Agencija za odgoj i obrazovanje

HRVATSKA ZAJEDNICA TEHNIČKE KULTURE

HRVATSKI SAVEZ INFORMATIČARA

Sastavni dio kompleta zadataka su i ove upute te uvodna stranica na kojoj se nalaze važni podatci o zadacima. Molimo vas da i jedno i drugo pažljivo pročitate. Na ostalim stranicama nalaze se tri zadatka. Prilikom rješavanja zadataka preporučuje se korištenje olovke i papira za skiciranje i razradu algoritma.

Prikupljanje i evaluacija rješenja, objavljivanje liste rezultata te rješavanje eventualnih žalbi dužnost je članova školskog povjerenstva te ste dužni slijediti njihove upute prije, tijekom te nakon završetka natjecanja. Članovi školskih povjerenstava evaluirat će vaša rješenja koristeći unaprijed pripremljene test podatke. Za svaki zadatak trebate predati izvorni kod rješenja te (u slučaju korištenja programskog jezika C/C++) i odgovarajuću izvršnu (exe) datoteku. Radi lakše i brže evaluacije, imena datoteka moraju odgovarati imenima zadataka. Primjerice, ako se zadatak zove "Neboder", predajte datoteke neboder.py ili neboder.cpp i neboder.exe.

Kod svakog pojedinog zadatka obratite pozornost na sekcije *Ulazni podatci* i *Izlazni podatci*. Tu su definirana pravila vezana uz format ulaznih i izlaznih podataka koji mora biti strogo poštovan kako bi vaša rješenja bila ispravno evaluirana. Vaš program sa standardnog ulaza mora očekivati samo zadane ulazne podatke, a na standardni izlaz ispisivati samo tražene izlazne podatke bez ikakvih dodatnih poruka. Vaši programi ne smiju pristupati nikakvim datotekama ili ih kreirati.

Prilikom rješavanja nekog zadatka i testiranja njegovog rješenja preporučuje se korištenje operatora redirekcije ulaza kako ne biste više puta nepotrebno unosili podatke preko tipkovnice. Na primjer, ulazne podatke za neki od oglednih primjera iz teksta zadatka možete spremiti u tekstualnu datoteku i testirati vaš program tako da ga pokrećete iz komandne linije na sljedeći način (pretpostavimo da se zadatak zove „Neboder"):

```
neboder.exe < primjer.txt
```

Znak < je operator redirekcije ulaza i sve što se nalazi u datoteci primjer.txt bit će proslijeđeno vašem programu kao da je uneseno preko tipkovnice.

Primjeri pravilno napisanih programa

Zadatak: Napišite program koji će zbrojiti i oduzeti dva cijela broja.

Ulaz: U prvom retku nalaze se dva cijela broja A i B, međusobno odvojena jednim razmakom.

Izlaz: U prvi redak ispišite zbroj, a u drugi redak razliku brojeva A i B.

C

```
#include <stdio.h>
int main(void) {
  int a, b;
  scanf("%d%d", &a, &b);
  printf("%d\n", a + b);
  printf("%d\n", a - b);
  return 0;
}
```

C++

```
#include <iostream>
using namespace std;
int main(void) {
  int a, b;
  cin >> a >> b;
  cout << a + b << endl;
  cout << a - b << endl;
  return 0;
}
```

Python 2

```
#!/usr/bin/python2
a, b = map(int, raw_input().split())
print a + b
print a - b
```

Python 3

```
#!/usr/bin/python3
a, b = map(int, input().split())
print(a + b)
print(a - b)
```

Za zadatke riješene u **Pythonu** potrebno je predati samo izvorni kod. Molimo da prva linija u kodu identificira inačicu Pythona koju treba koristiti kao u gornjim primjerima. Prilikom testiranja iz komandne linije potrebno je eksplicitno pozvati odgovarajući prevoditelj. Na primjer:

```
C:\Python27\python neboder.py < primjer.txt
```

U svom laboratoriju Dubravka treba odmjeriti N mililitara otopine, ali ne može pronaći menzuru. Na raspolaganju su joj tri epruvete za koje zna da su im volumeni redom A mililitara, B mililitara i C mililitara, a u njima se već nalaze poznate količine otopine (različite od N mL) i to redom K mililitara, L mililitara i M mililitara. To je sva otopina kojom Dubravka raspolaže.

Vaš je zadatak pomoći Dubravki i utvrditi može li ona u jednoj od epruveta dobiti točno N mililitara otopine, koristeći samo **jedno pretakanje** otopine iz jedne epruvete u drugu dok se prva ne isprazni ili druga ne napuni.

Evo primjera: pretpostavimo da treba dobiti 4 mL otopine koristeći epruvete volumena 3 mL, 5 mL i 8 mL, ispunjene s 2 mL, 5 mL i 0 mL, redom. Dovoljno je iz druge epruvete (5 mL) dopuniti prvu epruvetu dok se ona cijela ne napuni (2 mL → 3 mL). Prebačena količina bit će 1 mL, pa će u drugoj epruveti ostati 4 mL otopine, a tu smo količinu htjeli dobiti.

Napišite program koji pronalazi traženo pretakanje ili utvrđuje da ono ne postoji.

ULAZNI PODATCI

U prvom retku nalazi se prirodan broj N ($1 \leq N \leq 20$), tražena količina otopine.

U drugom retku nalaze se prirodni brojevi A, B i C ($1 \leq A, B, C \leq 20$) odvojeni razmakom, volumeni epruveta označenih rednim brojevima 1, 2, i 3 od kojih će barem jedan biti veći ili jednak N.

U drugom retku nalaze se cijeli brojevi K ($0 \leq K \leq A$), L ($0 \leq L \leq B$) i M ($0 \leq M \leq C$) odvojeni razmakom, trenutačne količine otopine u epruvetama 1, 2 i 3 redom, različite od N.

IZLAZNI PODATCI

Ako zadatak nije moguće obaviti, ispišite samo broj 0.

Inače, ispišite dva broja odvojena razmakom: najprije redni broj epruvete **iz** koje prelijevamo, a potom redni broj epruvete **u** koju prelijevamo otopinu dok se prva ne isprazni ili druga ne napuni, tako da nakon pretakanja u nekoj od epruveta bude točno N mL otopine. Test podatci bit će takvi da će traženo pretakanje (ako postoji) biti jedinstveno.

PRIMJERI TEST PODATAKA

ulaz

```
4
3 5 8
2 5 0
```

izlaz

```
2 1
```

ulaz

```
4
3 3 5
1 2 2
```

izlaz

```
2 3
```

ulaz

```
6
3 5 8
0 0 8
```

izlaz

```
0
```

Pojašnjenje prvog primjera: vidi tekst zadatka.

Pojašnjenje drugog primjera: pretakanjem sadržaja druge epruvete (2 mL) u treću epruvetu (2 mL), u trećoj će se naći 4 mL otopine.

Mutimir gradi stube od lego-kocaka. Njegove stube možemo prikazati kao niz stupaca od kojih svaki ima neku visinu izraženu brojem kocaka. Tako, primjerice, niz visina (1, 3, 4, 7) odgovara **strogo rastućim** stubama s lijeva na desno, a niz visina (8, 7, 4, 2) odgovara **strogo padajućim** stubama s lijeva na desno. S druge strane, nizovi visina (1, 2, 1, 4), (1, 2, 2, 4) i (4, 2, 2, 1) nisu ni strogo rastući ni strogo padajući.

Mutimir želi da njegove stube budu najprije strogo rastuće, a potom strogo padajuće. Preciznije, on želi da mu stube čine niz ($a_1, a_2, ..., a_N$) koji je strogo rastući do svojeg najvišeg, K-tog elementa a_K ($1 < K < N$), a nakon tog elementa strogo padajući. Njegov trenutačni niz stupaca ne zadovoljava to svojstvo i Mutimir stoga mora **dodati** neke kocke na neke stupce.

Evo primjera: ako Mutimir trenutačno ima niz stupaca s visinama (1, 1, 1, 1, 1), on može dodati po jednu kocku na 2. i 4. stupac te dvije kocke na 3. stupac, dobivajući tako niz (1, 2, 3, 2, 1) koji zadovoljava gornje svojstvo. Uvjet je moguće zadovoljiti i na druge načine, npr. dobiti niz (1, 4, 3, 2, 1) ili (1, 2, 3, 4, 1), no za takve stube potrebno je dodati više kocaka.

Pomozite Mutimiru i napišite program koji pronalazi **najmanji** ukupan broj kocaka koje treba dodati na stupce njegovog trenutačnog niza da bi dobiveni niz stuba zadovoljio opisano svojstvo „raste pa pada“.

(Obratite pažnju na sekciju Bodovanje).

ULAZNI PODATCI

U prvom retku nalazi se prirodan broj N ($1 \leq N \leq 20$), broj stupaca u Mutimirovom nizu.

U drugom retku nalazi se N prirodnih brojeva $a_1, a_2, ..., a_N$ ($1 \leq a_i \leq 100$) odvojeni razmakom, visine stupaca u nizu izražene brojem kocaka.

IZLAZNI PODATCI

U jedini redak ispišite traženi najmanji broj dodanih kocaka.

BODOVANJE

U test podatcima vrijednima 50% bodova, broj N bit će neparan i u optimalnom rješenju najviši stupac bit će točno na **sredini** niza (na poziciji (N+1)/2), kao u primjeru iz teksta zadatka.

PRIMJERI TEST PODATAKA

ulaz

```
5
1 1 1 1 1
```

izlaz

```
4
```

ulaz

```
7
3 6 5 1 9 4 7
```

izlaz

```
13
```

Pojašnjenje prvog primjera: vidi tekst zadatka (dodane su ukupno četiri kocke).

Pojašnjenje drugog primjera: konačni niz je (3, 6, 7, 8, 9, 8, 7).

Figura u igri sličnoj Tetrisu pada na donji dio ekrana (gdje su već pale neke figure). Figura koja pada je „obrnuti histogram“, tj. sastoji se od nekoliko stupaca poravnatih s gornje strane. Ekran prikazujemo tablicom od R redova i S stupaca, pri čemu je polje figure predstavljeno znakom '#' (hash ili ljestve), a prazan prostor znakom '.' (točka). Zadan je izgled ekrana prije padanja figure, tako da se u gornjem dijelu ekrana nalazi figura, kao u sljedećem primjeru:

```
..###..
..##...
..#....
.......
......#
..#...#
#######
```

Prije nego što figura počne padati, igrač je može pomicati lijevo i desno, ali nakon što počne padati, igrač je više ne može micati. Čim neki dio figure padne na neko popunjeno polje donjeg dijela ekrana, figura prestaje padati. Za gornji primjer (bez prethodnog pomicanja figure lijevo ili desno) nakon pada figure rezultat je sljedeći:

```
.......
.......
..###..
..##...
..#...#
..#...#
#######
```

Prazna polja ekrana **iznad kojih** se nađe **dio figure** koja je upravo pala zovemo **rupama**. U ovom slučaju dobili smo pet rupa, dolje radi ilustracije označenih znakovima '*':

```
.......
.......
..###..
..##*..
..#**.#
..#**.#
#######
```

Da smo figuru prije pada pomaknuli sasvim desno, dobili bismo samo jednu rupu (dolje neoznačenu):

```
.......
.......
.......
....###
....###
..#.#.#
#######
```

Napišite program koji unosi izgled ekrana prije pada figure te ispisuje izgled ekrana s najmanjim mogućim brojem rupa nakon pada figure.

(Obratite pažnju na sekciju Bodovanje.)

ULAZNI PODATCI

U prvom retku nalaze se prirodni brojevi R i S ($3 \leq R, S \leq 10$), dimenzije ekrana.

U sljedećih R redaka nalazi se po S znakova iz skupa {'.', '#'} koji opisuju izgled ekrana kao u tekstu zadatka, zadovoljavajući sljedeća svojstva:

- U gornjih nekoliko redaka nalazit će se figura, sačinjena od znakova '#', koja će zauzimati dva ili više uzastopnih stupaca od kojih će svaki biti popunjen od prvog retka niže. Drugim riječima, svi stupci figure započinjat će u prvom retku i neće sadržavati „rupe".
- Ispod figure nalazit će se barem jedan redak praznih polja, tj. znakova '.' (točka).
- Svako popunjeno polje (znak '#'), ako nije dio figure, bit će duž svog stupca povezan s donjim retkom ekrana, koji će biti cijeli popunjen. Tako ni u donjem dijelu ekrana neće biti „rupa".

IZLAZNI PODATCI

Ispišite izgled ekrana nakon padanja figure s najmanjim brojem dobivenih rupa, u istom obliku i istih dimenzija kao u ulaznim podatcima (R x S), koristeći iste znakove. Dakle, dobivene rupe trebaju ostati prikazane točkama, tj. znakovima '.' (u tekstu je korišten znak '*' samo radi pojašnjenja).

Test podatci bit će takvi da će rješenje biti jedinstveno, tj. postojat će samo jedan položaj figure koji rezultira najmanjim brojem rupa.

BODOVANJE

U test podatcima vrijednima 50% bodova figura će zauzimati sve stupce (cijelu širinu ekrana), što znači da je neće biti moguće micati lijevo ili desno (vidi prvi primjer ispod).

PRIMJERI TEST PODATAKA

ulaz

```
8 5
#####
.###.
..#..
.....
.....
..#..
.###.
#####
```

izlaz

```
.....
.....
#####
.###.
..#..
..#..
.###.
#####
```

ulaz

```
7 7
..###..
..##...
..#....
.......
......#
..#...#
#######
```

izlaz

```
.......
.......
.......
....###
....###
..#.#.#
#######
```