18. siječnja 2018.

2018 Natjecanje iz informatike

Školska razina 2018/ Osnovna škola (5. i 6. i 7. i 8. razred)

Primjena algoritama OŠ

OPISI ALGORITAMA

Agencija za odgoj i obrazovanje
Education and Teacher Training Agency

HRVATSKI SAVEZ INFORMATIČARA

Ministarstvo znanosti, obrazovanja i sporta

HRVATSKA ZAJEDNICA TEHNIČKE KULTURE

5.1. Zadatak: Težina — Autor: Nikola Dmitrović

Tražena vrijednost „nešto“ iz teksta zadatka jednaka je razlici zadane težine T i vrijednosti X.

Programski kod (pisan u Python 3)

```
T = int(input())
X = int(input())
print(T - X)
```

Potrebno znanje: naredba učitavanja, operator oduzimanja

Kategorija: ad hoc

5.2. Zadatak: Q5 — Autor: Nikola Dmitrović

Kako je oznaka za jučer „-1“, a oznaka za sutra „1“ čini se da je rješenje zadatka zbroj datuma D i oznake smjera S.

Programski kod (pisan u Python 3)

```
D = int(input())
S = int(input())
print(D + S)
```

Međutim, to je rješenje za koje ćemo dobiti 56 od 70 bodova. Iz trećeg primjera test podataka vidimo da to nije opće rješenje jer postoji slučaj kada iz 1. siječnja želimo otputovati u „jučer“ što je 31. prosinca. Ovaj primjer trebao je pomoći da se sjetimo da postoji još jedan poseban slučaj kada iz 31. siječnja želimo otići „sutra“ u 1. veljače. Kod implementacije ovih slučajeva trebamo biti oprezni kako se ne bi dogodili višestruki ispisi.

Programski kod (pisan u Python 3)

```
D = int(input())
S = int(input())
if D == 1 and S == -1:
    print(31)
elif D == 31 and S == 1:
    print(1)
else:
    print(D + S)
```

Potrebno znanje: naredba odlučivanja (osnovni i elif oblik)

Kategorija: ad hoc

5.3. Zadatak: London

Autor: Nikola Dmitrović

Ukupno trajanje puta od Zagreba do Frankfurta zbroj je duljine trajanja leta (TL) i kašnjenja (KL). Kako je avion krenuo na let točno u ponoć i na prelazak s leta na let se ne troši dodatno vrijeme samo trebamo provjeriti u kakvom su odnosu to vrijeme i vremena polijetanja aviona iz Frankfurta. To možemo napraviti korištenjem naredbe odlučivanja sa složenim logičkim uvjetom ili naredbe elif.

Programski kod (pisan u Python 3)

```
TL = int(input())
KL = int(input())
X = int(input())
Y = int(input())
Z = int(input())

let = TL + KL

if let <= X:
    print(1)
elif let <= Y:
    print(2)
else:
    print(3)
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

6.1. Zadatak: Q6 Autor: Nikola Dmitrović

Zadatak je sličan zadatku Q5 samo što se umjesto oznaka „1“ i „-1“ oznaka smjera putovanja zadaje znakovima „J“ i „S“.

Programski kod (pisan u Python 3)

```
D = int(input())
Z = input()

if Z == 'J':
    if D == 1:
        print(31)
    else:
        print(D - 1)

if Z == 'S':
    if D == 31:
        print(1)
    else:
        print(D + 1)
```

Potrebno znanje: učitavanje znakova, naredba odlučivanja

Kategorija: ad hoc

6.2. Zadatak: Ispit Autor: Nikola Dmitrović

Zadatak je podijeljen na dva dijela. Pri dio zadatka je odgovor na pitanje koja se ocjena dobila na testu. To ćemo lako odrediti tako što ćemo usporediti broj bodova N s rubnim vrijednostima za svaku od ocjena (45, 60, 75, 90). U drugom dijelu tražimo koliko bodova nedostaje do željene ocjene. To ćemo odrediti tako da ćemo od odgovarajuće rubne vrijednost oduzeti zadani broj N. Pri tome trebamo paziti na slučaj kada je ocjena koju smo dobili već veća ili jednaka od željene ocjene X kada trebamo ispisati riječ „REBEKA“.

Programski kod (pisan u Python 3)

```
N = int(input())
X = int(input())

if N < 45: ocjena = 1
elif N < 60: ocjena = 2
elif N < 75: ocjena = 3
elif N < 90: ocjena = 4
else: ocjena = 5
```

```
print(ocjena)


if X <= ocjena:
    print("REBEKA")
else:
    if X == 2:
        print(45 - N)
    if X == 3:
        print(60 - N)
    if X == 4:
        print(75 - N)
    if X == 5:
        print(90 - N)
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

6.3. Zadatak: Real — Autor: Nikola Dmitrović

Tekst zadatka je napisan u obliku natuknica koje su mogle pomoći da se zadatak lakše riješi i implementira rješenje. Dovoljno je samo provjeriti one slučajeve koji se spominju u zadatku, a ostali se mogu ignorirati jer ne utječu na rezultat.

Kako su ulazni podaci koji opisuju jednu utakmicu (Rp, Pp, Rk, Pk) zadani u jednom retku kao takve ih trebamo i učitati. U Pythonu ih možemo učitati kao listu (niz) ili redom u 4 varijable koristeći naredbu *map*.

Programski kod (pisan u Python 3)

```
N = int(input())
prosuo = 0


for i in range(N):
    Rp, Pp, Rk, Pk = map(int, input().split())
    if Rp > Pp:
        if Rk < Pk:
            prosuo += 3
        if Rk == Pk:
            prosuo += 2
    if Rp == Pp and Rk < Pk:
        prosuo += 1
print(prosuo)
```

Potrebno znanje: naredba učitavanja više elemenata u jednom retku, naredba odlučivanja, naredba ponavljanja

Kategorija: ad hoc

7.1. Zadatak: Lino — Autor: Nikola Dmitrović

Prvi redak ispisa možemo odrediti tako da prebrojimo koliko znakova „H“ ima u zadanom stringu. Za drugi redak trebamo pratiti trenutni rezultat i tražiti najveću vrijednost razlike.

Programski kod (pisan u Python 3)

```
N = int(input())
ishod = input()
razlika = H = S = 0
for i in ishod:
    if i == "H":
        H += 1
    else:
        S += 1
    if abs(H - S) > razlika:
        razlika = abs(H - S)
print(H)
print(razlika)
```

Potrebno znanje: string, algoritam traženja najvećeg elementa

Kategorija: ad hoc

7.2. Zadatak: Tablica — Autor: Stjepan Požgaj

U ovom zadatku moramo smisliti način na koji ćemo ukrasiti tablicu. Naravno, postoje razne ideje, no mi ćemo objasniti onu koja je po našem mišljenju najjednostavnija.

Definirajmo na početku varijable gore = 0, dolje = N - 1, lijevo = 0 i desno = M - 1.

Na početku cijelu matricu, tj. sva polja u pravokutniku kojem je gornja granica gore, donja dolje, lijeva lijevo i desna desno ispunimo zvjezdicama. Sada varijable gore i lijevo povećamo za 1, a varijable dolje i desno smanjimo za jedan. Nakon toga sva polja unutar ovih novih granica ispunimo slovima M. Pa opet promijenimo granice.

Postupak nastavljamo tako dugo dok je gore <= dolje i lijevo <= desno.

Programski kod (pisan u Python 3)

```
znak = "*M"
n, m = map(int, input().split())
mat = [['x'] * m for i in range(n)]
vrsta = 0
```

```
gore = 0
dolje = n - 1
lijevo = 0
desno = m - 1
while gore <= dolje and lijevo <= desno:
    for i in range(gore, dolje+1):
        for j in range(lijevo, desno+1):
            mat[i][j] = znak[vrsta]
    vrsta = (vrsta + 1) % 2
    gore += 1
    dolje -= 1
    lijevo += 1
    desno -= 1
for i in mat:
    print(''.join(i))
```

Potrebno znanje: dvodimenzionalni niz (lista lista)

Kategorija: ad hoc

7.3. Zadatak: Akcija — Autor: Nikola Dmitrović

Optimalnu podjelu N odabranih proizvoda u grupe s po K elemenata postižemo tako da cijene odabranih proizvoda prvo poredamo po veličini, od najskupljeg prema najjeftinijem i onda prvih K elemenata stavimo u prvu skupinu, drugih K u drugu. Elemente koji nisu mogli biti razvrstani u skupine ostavimo ili prebacimo u posebnu skupinu. Na kraju iz svake podskupine izbacimo proizvod s minimalnom cijenom, a sve ostale cijene zbrojimo.

Programski kod (pisan u Python 3)

```
N = int(input())
K = int(input())
# učitavanje cijena igračaka
igracke = []
for i in range(N):
    igracke += [int(input())]
# sortiranje
igracke = sorted(igracke)
igracke = igracke[::-1]
# podjela u skupine
skupine = []
for i in range(N // K):
```

```
    skupine += [igracke[:K]]
    igracke = igracke[K:]
# zbrajanje svih cijena minus najmanja
ukupno = sum(igracke)
for i in skupine:
    ukupno += sum(i) - min(i)
print(ukupno)
```

Potrebno znanje: liste (nizovi), sortiranje, traženje minimalnog elementa

Kategorija: ad hoc

8.1. Zadatak: Test — Autor: Nikola Dmitrović

U listu/niz *zna* treba učitati oznake svih zadataka koji Josip zna riješiti, a u drugu *blic* oznake zadataka koje je profesorica odabrala za test. Za svaki element iz liste *blic* trebamo provjeriti nalazi li se u listi zna. Na taj ćemo prebrojiti koliko će zadataka Josip riješiti na testu. Kod ispisa treba paziti na slučaj kada je Josip riješio nula zadataka.

Programski kod (pisan u Python 3)

```
N = int(input())
K = int(input())
zna = []
for i in range(K):
    zna += [int(input())]

blic = []
for i in range(5):
    blic += [int(input())]

ocjena = 0
for i in blic:
    if i in zna:
        ocjena += 1
if ocjena:
    print(ocjena)
else:
    print(1)
```

Potrebno znanje: lista

Kategorija: ad hoc

8.2. Zadatak: Šah

Autor: Marin Kišić

Iz danih podataka napravit ćemo tablicu u kojoj će u i-tom retku i j-tom stupcu pisati 0 ako se na toj poziciji ne nalazi niti jedna figura, 1 ako se na toj poziciji nalazi bijeli pijun ili 2 ako se na toj poziciji nalazi crni pijun. Nakon toga prolazit ćemo tablicom i kada dođemo do polja na kojem se nalazi bijeli pijun, provjerit ćemo nalazi li se crni pijun na polju (i-1, j-1). Ako se nalazi, povećat ćemo rješenje za 1 i označiti polje (i-1,j-1) s 0 da ne bismo tog pijuna kasnije brojali još jednom. Istu provjeru napravit ćemo za polje (i-1, j+1) te na kraju nakon što smo prošli sva polja ispisati rješenje.

Programski kod (pisan u C++)

```
#include <cstdio>
int n, m, x, y, p[10][10];

int main() {
  scanf("%d",&n);
  for (int i = 0; i < n; i++) {
    scanf("%d %d",&x,&y);
    p[x][y] = 1;
  }
  scanf("%d",&m);
  for (int i = 0; i < m; i++) {
    scanf("%d %d",&x,&y);
    p[x][y] = 2;
  }
  int rjesenje = 0;
  for (int i = 1; i <= 8; i++) {
    for (int j = 1; j <= 8; j++) {
      if (p[i][j] == 1 && p[i - 1][j - 1] == 2) {
        rjesenje++;
        p[i - 1][j - 1] = 0;
      }
      if (p[i][j] == 1 && p[i - 1][j + 1] == 2) {
        rjesenje++;
        p[i - 1][j + 1] = 0;
      }
    }
  }
  printf("%d\n",rjesenje);
  return 0;
```

```
}
```

Potrebno znanje: dvodimenzionalno polje (lista lista), naredba ponavljanja

Kategorija: ad hoc

8.3. Zadatak: Drvored Autor: Vedran Kurdija

Promotrimo slučaj za 70% bodova.

Konačno stanje drvoreda bit će: sva slova H, sva slova B te potom sva slova J. Ovo je lako konstruirati tako da prebrojimo koliko u početnom stanju ima kojih slova.

Ako se na nekoj poziciji nalazi ista vrsta stabla i u početnom i u završnom stanju drvoreda, tu poziciju nam se ne isplati dirati.

Pozicije na kojima se u početnom i u završnom stanju drvoreda nalazi različito stablo su pozicije na kojima ćemo morati promijeniti stabla. Lako zaključujemo da ćemo morati svako od stabala s tih pozicija izvaditi te ih zatim prerasporediti.

Iz toga je jasno da ćemo za svaku od tih pozicija na kojima se početno i završno stanje drvoreda razlikuju morati potrošiti dvije minute, jednu za vađenje stabla koje je na početku ondje, a drugu za stavljanje ispravnog stabla koje smo izvadili s neke druge pozicije na kojoj mu nije bilo mjesto.

Rješenje je, dakle, broj razlika početnog i završnog drvoreda * 2.

U rješenju za sve bodove nam valja provjeriti koliko nam minuta treba za sva moguća konačna stanja drvoreda (ima ih 6: HJB, HBJ, JBH, JHB, BHJ, BJH) te odaberemo ono stanje za koje nam treba najmanje minuta.

Programski kod (pisan u Python 3)

```
def rijesi(s1, s2, s3):
    global drvored, n
    konacni = ""
    for i in drvored:
        if i == s1:
            konacni += s1
    for i in drvored:
        if i == s2:
            konacni += s2
    for i in drvored:
        if i == s3:
            konacni += s3
    broj_razlika = 0
    for i in range(n):
        if drvored[i] != konacni[i]:
            broj_razlika += 1
```

```
    return broj_razlika * 2


drvored = input()
n = len(drvored)
slova = ['H', 'B', 'J']
rjesenje = 2 * n
for i in slova:
    for j in slova:
        for k in slova:
            if i != j and j != k and i != k:
                rjesenje = min(rjesenje, rijesi(i, j, k))
print(rjesenje)
```

Potrebno znanje: for petlja, stringovi

Kategorija: ad hoc