25. siječnja 2019.

2019 Natjecanje iz informatike

Školska razina 2019/ 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

5.1. Zadatak: Prsti — Autor: Nikola Dmitrović

Ako je zbroj dva zadana broja manji ili jednak od deset, tada trebamo ispisati taj zbroj. U protivnom ćemo uvijek ispisati vrijednost deset.

Programski kod (pisan u Python 3)

```
A = int(input())
B = int(input())
if A + B <= 10:
    print(A + B)
else:
    print(10)
```

Potrebno znanje: naredba učitavanja i ispisivanja, jednostavna naredba odlučivanja

Kategorija: ad hoc

5.2. Zadatak: Torta — Autor: Nikola Dmitrović

Za složiti jednu tortu prvo je potrebno nabaviti J cijelih jaja. Da bi dobili još Z žumanjaka i B bjelanjaka trebamo nabaviti ili Z cijelih jaja ili B cijelih jaja, ovisno o tome koji je od ta dva broja veći. Za napraviti N torti dovoljno je broj cijelih jaja potrebnih za jednu tortu pomnožiti s N.

Programski kod (pisan u Python 3)

```
N = int(input())
J = int(input())
Z = int(input())
B = int(input())
print(N * (J + max(Z, B)))
```

Potrebno znanje: naredba učitavanja i ispisivanja, jednostavna naredba odlučivanja

Kategorija: ad hoc

5.3. Zadatak: Bicikl — Autor: Nikola Dmitrović

Koristeći složenu naredbu odlučivanja trebamo raspisati uvjete definirane u zadatku i odrediti odgovarajuće ispise.

Programski kod (pisan u Python 3)

```
V = int(input())
if V <= 155:
    print('XS')
elif V <= 165:
    print('S')
elif V <= 175:
```

```
    print('M')
elif V <= 183:
    print('L')
elif V <= 191:
    print('XL')
else:
    print('XXL')
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

6.1. Zadatak: Braća Autor: Nikola Dmitrović

Koristeći naredbu odlučivanja raspisat ćemo svih 6 slučajeva koji se mogu dogoditi. Alternativno rješenje naslanja se na opservaciju da je 'VEDRAN' rješenje svaki put kada je Vedran jedan od dvojice braće, a 'MARIN' u svim ostalim slučajevima. Stjepan, kao najstariji brat, nikad neće biti rješenje.

Programski kod (pisan u Python 3)

```
T B1 = input()
B2 = input()
if B1 = 'M' and B2 = 'S': print('MARIN')
if B1 = 'M' and B2 = 'V': print('VEDRAN')
if B1 = 'V' and B2 = 'M': print('VEDRAN')
if B1 = 'V' and B2 = 'S': print('VEDRAN')
if B1 = 'S' and B2 = 'M': print('MARIN')
if B1 = 'S' and B2 = 'V': print('VEDRAN')
```

 # alternativno rješenje

```
B1 = input()
B2 = input()
if B1 == "V" or B2 == "V":
      print("VEDRAN")
else:
      print("MARIN")
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

6.2. Zadatak: Josip Autor: Nikola Dmitrović

U slučaju kada je J=1, zadatak smo mogli riješiti tako da prvo učitamo N, ukupan broj natjecatelja, te B1, broj bodova koje Josip ima. Koristeći naredbu ponavljanja slijedno ćemo učitavati sve ostale vrijednosti te prebrajati koliko ih je strogo veće od B1.

Općenito, sve vrijednosti učitamo u listu te prebrojimo koliko je vrijednosti strogo veće od broja bodova koji odgovara Josipovim bodovima.

Programski kod (pisan u Python 3)

```
N = int(input())
J = int(input())
bodovi = []
for i in range(N):
    bodovi += [int(input())]
koliko = 0
for i in range(N):
    if bodovi[i] > bodovi[J-1]: # J-1 zbog pomaka indeksiranja u listi
        koliko += 1
print(koliko)
```

Potrebno znanje: lista, naredba ponavljanja, prebrajanje

Kategorija: ad hoc

6.3. Zadatak: Simpsoni — Autor: Nikola Dmitrović

Prednost prvog bacanja odredit ćemo jednostavnom provjerom je li prvi učitani broj paran ili neparan, tj. je li djeljiv s dva ili nije.

Za određivanje broja osvojenih rundi i broja osvojenih bodova ključno je praćenje igrača na potezu. Igrač na potezu mijenja se svaki put kada su brojevi dobiveni na kockicama jednaki. Ovisno o tome tko je na potezu i odnosa prvog i drugog dobivenog broja povećavamo broj osvojenih rundi i dobivenih bodova.

Programski kod (pisan u Python 3)

```
P = int(input())
N = int(input())
# prednost prvog bacanja, paran-Kang-0, neparan-Konos-1
if P % 2 == 0:
    prednost = 0
    print('KANG')
else:
    prednost = 1
    print('KONOS')
kangR = kangB = konosR = konosB = 0
for i in range(N):
    K1, K2 = map(int, input().split())
    if K1 > K2:
```

```
        if prednost == 0:
            kangR += 1
            kangB += K1 + K2
        else:
            konosR += 1
            konosB += K1 + K2
    elif K2 > K1:
        if prednost == 1:
            kangR += 1
            kangB += K1 + K2
        else:
            konosR += 1
            konosB += K1 + K2
    else:
        if prednost == 0:
            prednost = 1
        else:
            prednost = 0
print(kangR, konosR)
print(kangB, konosB)
```

Potrebno znanje: naredba ponavljanja

Kategorija: ad hoc

7.1. Zadatak: Nula Autor: Nikola Dmitrović

Zadani niz brojeva učitat ćemo u listu, tj. niz. Za svaki element liste koji je nula, a nalazi se između drugog i predzadnjeg mjesta u listi, provjerit ćemo koja se vrijednost nalazi jedno mjesto prije, a koja jedno mjesto poslije. Ovisno o ispunjenosti uvjeta opisanih u zadatku prebrojit ćemo broj pozitivnih i negativnih nula.

Programski kod (pisan u Python 3)

```
N = int(input())
brojevi = []
for i in range(N):
    brojevi += [int(input())]
pozitivna = negativna = 0
for i in range(1,N-1):
    if brojevi[i] == 0:
        if brojevi[i-1] < 0 and brojevi[i+1] > 0: pozitivna += 1
```

```
        if brojevi[i-1] > 0 and brojevi[i+1] < 0: negativna += 1
print(pozitivna, negativna)
```

Potrebno znanje: lista, naredba ponavljanja, naredba odlučivanja

Kategorija: ad hoc

7.2. Zadatak: Picigin — Autor: Nikola Dmitrović

Rekonstruirat ćemo igru unatrag analizom odigranih poteza od posljednjeg prema prvom. Naravno, rekonstrukcija igre unatrag zahtjeva i odgovarajuću promjenu pravila igre.

Programski kod (pisan u Python 3)

```
def smjer(potez, x):
    if x == 1:
        if potez == 1: return 4
        if potez == 2: return 1
        if potez == 3: return 2
        if potez == 4: return 3
    if x == -1:
        if potez == 1: return 2
        if potez == 2: return 3
        if potez == 3: return 4
        if potez == 4: return 1
    if x == 0:
        if potez == 1: return 3
        if potez == 2: return 4
        if potez == 3: return 1
        if potez == 4: return 2
N = int(input())
dodavanja = []
for i in range(N):
    dodavanja += [int(input())]
dodavanja = dodavanja[::-1]
potez = int(input())
for i in range(N):
    potez = smjer(potez, dodavanja[i])
print(potez)
```

Potrebno znanje: lista

Kategorija: simulacija

7.3. Zadatak: Uber — Autor: Nikola Dmitrović

Za svaki kamion s popisa provjerit ćemo kojem od tri slučaja pripada i poduzeti odgovarajuće akcije. Parkiralište ćemo simulirati jednom listom od M elemenata, a mjesta koja kamion zauzima upisivanjem njegove oznake na pripadajuće mjesto.

Programski kod (pisan u Python 3)

```
N, M, T = map(int, input().split())
duljina = list(map(int, input().split()))
signali = list(map(int, input().split()))
parkiraliste = [0] * M
vrijeme = 0
for kamion in signali:
    if kamion in parkiraliste:
        parkiraliste = [0 if x == kamion else x for x in parkiraliste]
        vrijeme += 1
    else:
        vrijeme += 2
        nasao = 0
        for i in range(M-duljina[kamion - 1]+1):
            if sum(parkiraliste[i:i+duljina[kamion-1]]) == 0:
                nasao = 1
                parkiraliste = [kamion if i <= index < i+duljina[kamion-1] else
x for index,x in enumerate(parkiraliste)]
                break
        if nasao == 0:
            vrijeme += 3
print(vrijeme)
```

Potrebno znanje: lista

Kategorija: simulacija

8.1. Zadatak: Josip — Autor: Nikola Dmitrović

Vidi zadatak 6.2.

8.2. Zadatak: Uber — Autor: Nikola Dmitrović

Vidi zadatak 7.3.

8.3. Zadatak: Narukvica — Autor: Mislav Bradač

Kako bi riješili zadatak iz prošlogodišnje narukvice ćemo generirati sve narukvice koje možemo dobiti s tri rezanja. To možemo napraviti pomoću tri petlje gdje svaka petlja određuje jednu poziciji na kojoj

ćemo prerezati prošlogodišnju narukvicu. U tijelu najunutarnije petlje konstruiramo tri niza znakova gdje je svaki od njih jednak jednom od dijelova nastalim rezanjem prošlogodišnje narukvice. Primjećujemo da sada možemo ta tri dijela zalijepiti u dva moguća redoslijeda: prvi, drugi pa treći ili prvi, treći pa drugi dio. Ostali redoslijedi su ciklički jednaki s jednim od ova dva navedena redoslijeda pa ih ne moramo promatrati. Za oba redoslijeda konstruiramo pripadne nizove znakova koji nam predstavljaju narukvice koje možemo dobiti postupkom iz zadatka. Još ostaje provjeriti je li neka od cikličkih rotacija niza jednaka ovogodišnjem kodu. To možemo učiniti tako da generiramo još jednom petljom sve rotacije pa usporedimo nizove znakova ili tako da dupliciramo niz znakova koji smo generirali, na način da na njegov kraj zalijepimo njega samoga, i provjerimo nalazili se ovogodišnji kod unutar dupliciranog niza.

Programski kod (pisan u Python 3)

```
Ta = input()
n = len(a)
a += a
m = int(input())
for _ in range(m):
    b = input()
    b += b
    found = False
    for i in range(n):
        for j in range(n + 1):
            for k in range(n + 1 - j):
                part1 = a[i : i + j]
                part2 = a[i + j : i + j + k]
                part3 = a[i + j + k : i + n]
                new_tape1 = part1 + part2 + part3
                new_tape2 = part1 + part3 + part2
                if b.find(new_tape1) != -1 or b.find(new_tape2) != -1:
                    found = True
    print("DA" if found else "NE")
```

Potrebno znanje: string

Kategorija: ad hoc