21. siječnja

Školsko natjecanje / Osnovna škola (5. i 6. i 7. i 8. razred)

Primjena algoritama (Basic/Python/Pascal/C/C++)

OPISI ALGORITAMA

Programski kod za svaki zadatak bit će napisan u programskom jeziku Python. Na taj način želimo učenike upoznati s novim jezikom te ih motivirati da se dodatno posvete njegovom proučavanju.

Agencija za odgoj i obrazovanje
Education and Teacher Training Agency

HRVATSKI SAVEZ INFORMATIČARA

Ministarstvo znanosti, obrazovanja i sporta

5.1. Zadatak: Bajadera — Autor: Nikola Dmitrović

Ako je Krešimir dobio N bajadera, a zatim jednu bajaderu dao tati, jednu mami i jednu sestri, očito je da je njemu ostalo N – 3 bajadera.

Programski kod (pisan u Pythonu 3.4)

```
N = int(input())
print(N - 3)
```

Potrebno znanje: naredba učitavanja i ispisivanja, operator oduzimanja

Kategorija: ad hoc

5.2. Zadatak: Igra — Autor: Adrian Satja Kurdija

Potrebno je unijeti brojeve A, B, C i D te ispisati sljedeće:

- A (broj bodova na prvoj razini),
- A + B (broj bodova na prvim dvjema razinama),
- A + B + C (broj bodova na prvim trima razinama),
- A + B + C + D (broj bodova na svim četirima razinama).

Potom za broj A + B + C + D (ukupan broj bodova) treba s pomoću if-naredbe provjeriti je li manji od 100. Ako jest, ispisujemo NE, a inače ispisujemo DA.

Programski kod (pisan u Pythonu 2.7)

```
a = input()
b = input()
c = input()
d = input()
print a
print a + b
print a + b + c
print a + b + c + d
if a + b + c + d >= 100:
    print 'DA'
else:
    print 'NE'
```

Potrebno znanje: osnovne operacije, if-then-else

Kategorija: ad hoc

5.3. Zadatak: Kockica — Autor: Nikola Dmitrović

Prvi dio zadatka:

U zadatku piše da je zbroj broja na strani koja je okrenuta prema gore i broja na strani okrenutoj prema dolje jednak 7. Iz te rečenice slijedi da je broj na strani okrenutoj prema gore jednak razlici broja 7 i broja na strani okrenutoj prema dolje (koja se zadaje u zadatku).

Drugi dio zadatka:

Okretanje kockice u desno za jednu četvrtinu kruga možemo simulirati tako da na komadiću papira nacrtamo originalni izgled i vrtimo u smjeru kazaljke na satu za jednu četvrtinu kruga. Za svaki od šest brojeva vrijedi sljedeće:

Original	Nakon okretanja	Original	Nakon okretanja	Original	Nakon okretanja
* * * * O * * * *	* * * * O * * * *	* * O * * * O * *	O * * * * * * * O	* * O * O * O * *	O * * * O * * * O
O * O * * * O * O	O * O * * * O * O	O * O * O * O * O	O * O * O * O * O	O * O O * O O * O	O O O * * * O O O

Programski kod (pisan u Pythonu 3.4)

```
N = int(input())
print(7 - N)
if N == 1: print("OOO","***","OOO", sep = "\n") # dolje 1, gore 6
if N == 2: print("O*O","*O*","O*O", sep = "\n") # dolje 2, gore 5
if N == 3: print("O*O","***","O*O", sep = "\n") # dolje 3, gore 4
if N == 4: print("O**","*O*","**O", sep = "\n") # dolje 4, gore 3
if N == 5: print("O**","***","**O", sep = "\n") # dolje 5, gore 2
if N == 6: print("***","*O*","***", sep = "\n") # dolje 6, gore 1
```

Potrebno znanje: naredba učitavanja, naredba ispisa, jednostavna naredba odlučivanja

Kategorija: ad hoc

6.1. Zadatak: Žabe — Autor: Adrian Satja Kurdija

Najprije unosimo brojeve A, B, C i D -- oznake žaba. Potom *If*-naredbom provjeravamo je li A paran. (Prisjetimo se: broj je paran ako mu je ostatak pri dijeljenju s dva jednak nula.) Ako je paran, ispisujemo D, A, B, C (rotacija udesno), a inače ispisujemo B, C, D, A (rotacija ulijevo).

Programski kod (pisan u Pythonu 2.7)

```
a = input()
b = input()
c = input()
d = input()
if int(a) % 2 == 0:
    print d, a, b, c
else:
    print b, c, d, a
```

Potrebno znanje: dijeljenje s ostatkom, if-then-else

Kategorija: ad hoc

6.2. Zadatak: Test Autor: Nikola Dmitrović

Koderski zadatak u kojem je nužno samo točno nakodirati uvjete zadane u zadatku. Za točno rješenje zadatka nužno je:

1. posebno zbrojiti redom sve učitane brojeve na neparnom mjestu i sve brojeve na parnom mjestu te istovremeno prebrojiti koliko je bilo učitanih na parnom mjestu, a koliko na neparnom mjestu;
2. odrediti prosjek učitanih brojeva;
3. ispisati zadane prosjeke pazeći da se sa službenim poklapa na prve dvije decimale;
4. naredbom odlučivanja odrediti i ispisati tražene poruke.

Programski kod (pisan u Pythonu 3.4)

```
N = int(input())
parni = neparni = parni_koliko = neparni_koliko = 0
for i in range(1, N + 1):
    O = int(input())
    if i % 2 == 1:
        neparni += O
        neparni_koliko += 1
    else:
        parni += O
        parni_koliko += 1

prosjek_parni = parni / parni_koliko
prosjek_neparni = neparni / neparni_koliko

print(round(prosjek_parni, 3))
print(round(prosjek_neparni,3))

if prosjek_parni > prosjek_neparni:
    print("P")
elif prosjek_neparni > prosjek_parni:
    print("N")
else:
    print("PN")
```

Potrebno znanje: naredba učitavanja i ispisivanja, naredba ponavljanja, naredba odlučivanja

Kategorija: ad hoc

6.3. Zadatak: Stranice

Autor: Adrian Satja Kurdija

Neka je **N** traženi broj stranica. Na početku stavljamo **N** = 0 i potom *while*-petljom dodajemo jednu po jednu stranicu -- dakle, unutar petlje povećavamo **N** za jedan. Pritom u pomoćnoj varijabli **dosadasnji_broj_znamenaka** pamtimo koliko smo znamenaka iskoristili za označavanje svih dosad dodanih stranica. Kada taj broj postane jednak zadanom broju znamenaka **Z**, prekidamo petlju i ispisujemo **N**, tj. broj stranica koje smo do tada označili.

Ovo je bila glavna ideja, a sada objasnimo najvažniji detalj gornjeg algoritma: kako računamo dosadašnji broj iskorištenih znamenaka? Jasno, taj je broj na početku nula, a unutar petlje valja ga povećati za broj znamenaka od **N**. A kako računamo koliko **N** ima znamenaka?

Dijeljenjem broja s 10 njegov broj znamenaka smanjuje se za jedan. Dakle, da bismo dobili broj znamenaka od **N**, trebalo bi ga *while*-petljom dijeliti s 10 dok god je veći od nula, i prilikom svakog dijeljenja povećati brojač znamenaka za jedan.

Tu moramo biti oprezni: ne smijemo dijeliti samu varijablu **N** jer ona treba ostati očuvana za daljnje korištenje. Taj problem lako rješavamo bilo stvaranjem kopije (vrijednost nove varijable **kopija_N** postavljamo na **N** i dijelimo nju umjesto **N**), bilo s pomoću funkcije koja prima prirodan broj i vraća broj njegovih znamenaka.

Na kraju napomenimo da postoji i drugi, nezgrapniji način određivanja broja znamenki od **N**, a taj je s pomoću *if*-naredbi -- uspoređujući **N** sa 10, 100, 1000, …, 1 000 000, koji ne preporučujemo zbog velikog broja slučajeva i veće mogućnosti pogreške.

Programski kod (pisan u Pythonu 2.7)

```
def broj_znamenaka(n):
    brz = 0
    while n > 0:
        brz += 1
        n /= 10
    return brz
z = input()
n = 0
trenutni_broj_znamenaka = 0
while trenutni_broj_znamenaka < z:
    n += 1
    trenutni_broj_znamenaka += broj_znamenaka(n)
print n
```

Potrebno znanje: while-petlja, rad sa znamenkama

Kategorija: ad hoc

7.1. Zadatak: Pruga — Autor: Nikola Dmitrović

Na svakoj od N postaja na pruzi možemo kupiti kartu do bilo koje od N -1 postaja na toj pruzi u kojoj trenutno nismo. Naravno da nećemo kupiti kartu do postaje u kojoj kupujemo kartu. Iz toga slijedi da je ukupan broj karata koje možemo kupiti N*(N-1).

Ako se netko nije sjetio formule, mogao je dvostrukom naredbom ponavljanja simulirati kupovinu karata tako da je za svaku stanicu obišao sve preostale stanice i povećao brojač za jedan.

Programski kod (pisan u Pythonu 3.4)

```
N = int(input())
print(N * (N - 1))
```

Potrebno znanje: naredba učitavanja i ispisivanja

Kategorija: ad hoc

7.2. Zadatak: Košarica — Autor: Nikola Dmitrović

Zadatak možemo riješiti tako da doslovno pratimo i implementiramo tekst zadatka. Prvo učitamo brojeve N i X. Zamislimo sada da smo došli na kraj reda. Jasno je da smo X + 1 u redu. Prvi sljedeći učitani broj je broj proizvoda u košarici osobe ispred nas. Ako je broj proizvoda u našoj košarici manji od učitanog broja, pomičemo se u redu jedno mjesto naprijed (X + 1 → X) i nastavljamo učitavati podatke. Ako nije, ostajemo na mjestu gdje jesmo i prestajemo učitavati podatke.

Zadatak **8.2. Košara** razlikuje se od Košarice po tome što se podaci zadaju počevši od košarice prvog kupca u redu. Zbog male promjene u obliku ulaznih podataka podatke moramo učitati u niz (listu) te zatim primijeniti opisani algoritam.

Programski kod (pisan u Pythonu 3.4)

```
N = int(input())
X = int(input())
po_redu = X + 1
for i in range(X):
    Pi = int(input())
    if N < Pi:
        po_redu -= 1
    else:
        break
print(po_redu)
```

Potrebno znanje: naredba učitavanja i ispisivanja, naredba ponavljanja, naredba odlučivanja

Kategorija: simulacija

7.3. Zadatak: Neron — Autor: Nikola Dmitrović

Ključnu riječ i zadani broj učitamo kao stringove. Svaki znak u broju zamijenimo pripadajućim slovom na odgovarajućem mjestu u ključnoj riječi. Primijetimo da dio zadatka o izbacivanju znaka X iz nove riječi slobodno možemo zanemariti jer ćemo prilikom kreiranja te riječi zanemariti sve znamenke koje su veće od duljine ključne riječi ili jednake nuli.

Programski kod (pisan u Pythonu 3.4)

```
kljucna = input()
N = input()
rijec = ""
for i in N:
    if int(i) <= len(kljucna) and int(i) != 0:
        rijec += kljucna[int(i) - 1]
print(rijec)
```

Potrebno znanje: rad sa stringovima

Kategorija: simulacija

8.1. Zadatak: Preciznost — Autor: Adrian Satja Kurdija

Da bismo provjerili razlikuju li se broj A + B i broj C za manje od 0.000001, valja pogledati je li njihova razlika po apsolutnoj vrijednosti manja od 0.000001. Dakle, ako je $|A + B - C| < 0.000001$, ispisujemo '=', a inače provjeravamo je li $A + B > C$. Ako jest, ispisujemo '>', a inače ispisujemo '<'.

Programski kod (pisan u Pythonu 2.7)

```
a = input()
b = input()
c = input()
if abs(a + b - c) < .000001:
    print '='
elif a + b < c:
    print '<'
else:
    print '>'
```

Potrebno znanje: if-then-else, apsolutna vrijednost, realni brojevi

Kategorija: ad hoc

8.2. Zadatak: Košara — Autor: Nikola Dmitrović

Zadatak možemo riješiti tako da doslovno pratimo i implementiramo tekst zadatka.

Prvo učitamo brojeve N i X. Kako su podaci u ulazu zadani od prve osobe u redu do posljednje osobe u redu, nije moguće u potpunosti slijediti algoritam iz zadatka 7.2. Košarica, već je nužno podatke prvo učitati u niz (listu). Zamislimo sada da smo došli na kraj reda (jedno mjesto iza zadnjeg elementa u nizu). Jasno je da smo X + 1 u redu. Zadnji broj u nizu je broj proizvoda u košarici osobe ispred nas. Ako je broj proizvoda u našoj košarici manji od tog broja, pomičemo se u redu jedno mjesto naprijed (X + 1 → X) i prelazimo na analizu predzadnjeg elementa u nizu. Ako nije, ostajemo u redu na mjestu gdje jesmo i prestajemo analizirati podatke.

Programski kod (pisan u Pythonu 3.4)

```
N = int(input())
X = int(input())
red = []
for i in range(X):
    red += [int(input())]
po_redu = X + 1
for i in range(X - 1, -1, -1):
    if N < red[i]:
        po_redu -= 1
    else:
        break
print(po_redu)
```

Potrebno znanje: naredba učitavanja i ispisivanja, niz, naredba ponavljanja i naredba odlučivanja

Kategorija: simulacija

8.3. Zadatak: Poredak — Autor: Adrian Satja Kurdija

Možemo učitati **N** kao prirodan broj, ali tada moramo "izvlačiti" njegove znamenke uzastopnim dijeljenjem s 10. Ovdje je lakše **N** promatrati kao string od pet znakova koje redom prolazimo for-petljom.

Unutar petlje najprije želimo od trenutnoga, **i**-tog znaka stringa **N** dobiti odgovarajući jednoznamenkasti broj (nazovimo ga **k**), npr. '4' → 4. To činimo tako da od ASCII vrijednosti dotičnoga znaka oduzmemo ASCII vrijednost znaka '0'. Potom koristimo glavni zaključak: ako na **i**-tom mjestu od **N** stoji znamenka **k**, onda na **k**-tom mjestu traženog rješenja mora stajati znamenka **i**. Dakle, u nizu koji predstavlja rješenje valja na **k**-to mjesto upisati **i**.

Naravno, na koncu ispisujemo niz znamenaka koji predstavlja rješenje, a popunili smo ga tijekom gore opisane petlje.

Programski kod (pisan u Pythonu 2.7)

```
n = raw_input()
m = [0 for i in range(5)]
for i in range(5):
    m[ord(n[i]) - ord('1')] = i + 1
print ''.join(map(str, m))
```

Potrebno znanje: for-petlja, stringovi

Kategorija: ad hoc