Zadatak 1

Pristup 1

Obrtanje redosleda cifara upotrebom stringova. Python funkcija za obrtanje stringa: ***str(n)[::-1]***

Pristup 2

Matematički pristup. Ideja:

- inicijalizovati rezultat na 0, npr. promenljiva ***reverse***
- preuzimati vrednost poslednje cifre broja ***lastDigit = number % 10***, operator % je ***moduo*** i predstavlja ostatak pri deljenju
- dodati poslednju cifru na kraj na promenljivu reverse, reverse = (reverse * 10) + lastDigit
- **u**kloniti poslednju cifru originalnog broja ***number = number / 10***

Zadatak 2

Pristup 1

Brute force: računanje proizvoda svih trocifrenih brojeva i traženje najvećeg palindroma.

Pristup 2

Rešenje bazirano na uočenoj simetriji matrice proizvoda. Na primer, imamo niz [1, 2, 3, 4, 5]. Ako želimo da svaki element pomnožimo sa svakim, računamo sledeće proizvode:

*1 * 1*	*1 * 2*	*1 * 3*	*1 * 4*	*1 * 5*
*2 * 1*	*2 * 2*	*2 * 3*	*2 * 4*	*2 * 5*
*3 * 1*	*3 * 2*	*3 * 3*	*3 * 4*	*3 * 5*
*4 * 4*	*4 * 2*	*4 * 3*	*4 * 4*	*4 * 5*
*5 * 4*	*5 * 2*	*5 * 3*	*5 * 4*	*5 * 5*

Primećeno je da se neki proizvodi ponavljaju. U svakom redu matrice, proizvodi pre indeksa trenutnog reda + 1 je već računat i ne treba ga ponovo računati i.

Pristup 3

Rešenje bazirano na zaključku da je palindrom deljiv sa 11.

Pristup 4

Pronalaženje najvećeg palindroma pretragom po ‘trakama'.

Zadatak 3

Fibonacci pretpostavke:

- F(1) = 1
- F(2) = 1
- F(n) = F(n-1) + F(n-2)
- početak niza: 1, 1, 2, 3, 5, 8, 13...

Pseudocode:

```
FOR i = 1 TO N DO
  IF (i = 1)
    F1 = 1
    WRITE (F1)
  ELSE IF (i = 2)
    F2 = 1
    WRITE (F2)
  ELSE
    F3 = F1 + F2
    WRITE (F3)
    F1 = F2      new_previous
    F2 = F3      new_current
  END IF-ELSE
END FOR
```

Pristup 1

Računati sve Fibonačijeve brojeve od početka i za svaki proveriti da li je paran, i ukoliko jeste dodati ga na zbir. Uslov za kraj računanja je kada suma brojeva bude veća od MAX_VALUE = 4000000.

Pristup 2

Rešenje bazirano na pretpostavci da je svaki treći element Fibonačijevog niza paran broj: **0**, 1, 1, **2**, 3, 5, **8**, 13, 21, **34**, 55, 89, **144**, 233, 377, **610**, 987, 1597, **2584**, 4181, 6765, **10946**, 17711, 28657, **46368**, 75025, 121393, **196418**, 317811

Kod ovog pristupa, ***new_previous*** i ***new_previous*** vrednosti se računaju na sledeći način:

- ***new_previous = previous + 2 * current***
- ***new_current = 2 * previous + 3 * current***

Previous će biti Fibonačijev broj koji je deljiva sa 2, a ***current*** će biti prvi sledeći posle njega.

Zadatak 4

Collatz conjecture

Pristup 1

Napraviti niz elemenata ***seq_lengths*** koji čuva dužine sekvenci za brojeve. Indeks niza odgovara broju za koji je izračunata sekvenca. Računati dužinu sekvence za svaki broj do MAX_VALUE = 1000000 tako što se u svakoj iteraciji izračunava ***n*** po specificiranoj formuli. Računanje za trenutni broj se prekida u dva slučaja:

- Ako je već izračunata dužina sekvence za broj ***n*** (broj ***n*** je manji od dužine niza), uvećati dužinu sekvence za ***seq_lengths[i] + 1***,
- Ako je ***n = 1***, uvećati dužinu sekvence za ***1***.

Kada se izračuna dužina sekvence za neki broj, doda se na kraj niza ***seq_lengths*** i prelazi na računanje dužine za sledeći broj.

Pristup 2

Rekurzivni pristup. Inicijalizovati niz ***seq_lengths*** tako da se popuni sa MAX_VALUE = 1000000. Postaviti prva dva broja na 0 i 1, jer su ovo poznate dužine sekvenci. Za svaki broj se poziva funkcija koja će za trenutni broj ***n*** računati dužinu sekvence i koja vraća 3 broja:

- ***1***, ukoliko je ***n = 1*** (uslov 1)
- ***seq_lengths[n]***, ukoliko je za broj n već izračunata dužina sekvence (***seq_lengths[n] != 0)*** (uslov 2)
- Izračunatu vrednost.

Ukoliko prva dva uslova nisu ispunjena (***n*** nije 1 i za ovaj broj nije izračunata sekvenca), trenutnu dužinu uvećaj za 1 + rekurzivni poziv funkcije. Trenutna dužina će se uvećavati za 1 sve dok neki od prva dva uslova ne budu zadovoljena.

Kada se izračuna dužina sekvence za neki broj, doda se na kraj niza ***seq_lengths,*** funckija vraća izračunatu vrednost i prelazi na računanje dužine za sledeći broj.

*** uslov 1 i uslov 2 su ***bazni uslovi***. Izvršavanje rekurzivne funkcije se prekida kada je neki od ovih uslova ispunje.

Zadatak 5

Pristup 1

Brute force: Iterirati kroz niz čiji su elementi ***1 do n*** i proveriti da li je broj ***n*** deljiv sa nekim brojem iz niza.

Pristup 2

Iterirati kroz niz čiji su elementi ***1 do √n*** i proveriti da li je broj ***n*** deljiv sa nekim brojem iz niza. Ako broj ***n*** rastavimo na faktore i ako su oba veća od korena broja, tada je njihov proizvod sigurno veći od broja ***n***, što znači da bar jedan faktor mora biti manji ili veći od korena. Ukoliko ne možemo pronaći neki faktor koji je manji ili veći od korena, broj je prost.
Python funkcija za računanje korena: ***n**0.5***

Pristup 3

Sieve of Eratosthenes

Pseudocode:

```
Let A be an array of Boolean values, indexed by
integers 2 to n, initially all set to true.
 for i = 2, 3, 4, ..., √n:
   if A[i] is true:
     for j = i², i²+i, i²+2i, i²+3i, ..., n:
       A[j] := false
```

Zadatak 6

Pristup 1

Najjednostavniji algoritam za pronalaženje najvećeg faktora je višestrukim deljenjem broja sa prostim faktorom dok broj ne postane 1. Početni prosti faktor je $f = 2$, ako je broj n deljiv sa f, podeliti broj n sa f, u suprotnom uvećati f za 1.

Zadatak 7

Pristup 1

- NZS dva broja a i b: ***a*b / nzd(a, b)***
- NZD dva broja a i b: dok važi uslov ***b != 0***, postaviti ***a = b***, ***b = a % b*** (% je moduo, ostatak pri deljenju). Kada je ***b = 0***, vrednost NZD je broj ***a***

Za svaki broj ***i*** od 1 do n izračunati NZS brojeva ***i*** i NZS koji je izračunat u prethodnoj iteraciji. NZS prve iteracije je ***1***.