Praktikum iz programiranja 3

2023/24

Primer

- U tekstu može da se nađe izraz oblika broj1! ili izraz broj1^broj2, gde su broj1 i broj2 celi brojevi, operacija ! je faktorijel, a ^ označava stepen. Ukoliko se u tekstu pojave izrazi ovog oblika, potrebno ih je zameniti rezultatom odgovarajućih operacija.

Ulaz	Izlaz
Temperatura za ponedeljak je 4!. Vazdusni pritisak se procenjuje na 2^10. Sansa za pojavu kise je !50%. Ponesite kisobran!	Temperatura za ponedeljak je 24. Vazdusni pritisak se procenjuje na 1024. Sansa za pojavu kise je !50%. Ponesite kisobran!

```
import re
import math as m
s = """Temperatura za ponedeljak je 4!.
Vazdusni pritisak se procenjuje na 2^10.
Sansa za pojavu kise je !50%. Ponesite kisobran!
"""

regex = re.compile(r'\d+!|\d+\^\d+')
L = regex.findall(s)
for x in L:
    if x.find("!")!=-1:
        br = m.factorial(int(x[:-1]))
        print(br)
        s = s.replace(x,str(br))
    else:
        poz = x.find("^")
        br1 = int(x[:poz])
        br2 = int(x[poz+1:])
        print(br1,br2)
        s = s.replace(x,str(br1**br2))
print(s)
```

Primer

- Napisati funkciju Bullet koja ispituje da li niz karakera, koji prihavata kao argument, predstavlja ispravnu liniju u nabrajanju. Ispravna linija u nabrajanju je ako
  - počinje brojem (bar jedna cifara),
  - potom se nalazi bar jedan od simbola slova, cifara i SPACE karaker u proizvoljnom redosledu i
  - završava se jednim od znakova „.“, „?“, „!“, iza kojih su dozvoljeni samo SPACE karakteri, ali mogu biti izostavljeni.

Ulaz	Izlaz
1Prva linija. 2 linija Treca linija! 55? 41. C!	1Prva linija. 55? 41. C!

```
import re
def Bullet(x):
    regex = re.compile(r'^(\d+)+([\w\d ]+)+([\.\?\!]+)+([ ]?)')
    L = regex.findall(x)
    if L==[]:
        return False
    else:
        return True
s = """1Prva linija.
2 linija
Treca linija!
55?
41. C!"""
L=s.split("\n")
print(L)

nova_L=[]
for x in L:
    if Bullet(x):
        nova_L = nova_L + [x]
print(nova_L)
```

```
['1Prva linija.', '2 linija', 'Treca linija!', '55? ', '41. C!']
['1Prva linija.', '55? ', '41. C!']
```

Najveći zajednički delilac (NZD) Euklidov algoritam

- Euklidov algoritam je efikasan način za određivanje najvećeg zajedničkog delioca (NZD) datih brojeva. NZD dva broja je najveći broj koji istovremeno deli oba bez ostatka.

- Euklidov algoritam je zasnovan na principu da se najveći zajednički delilac dva broja ne menja ukoliko se manji broj oduzme od većeg, pa se zatim odredi NZD novodobijenog broja i manjeg od prethodna dva.

- Na primer, imamo brojeve 48 i 18. Ukoliko od 48 oduzmemo 18 dobićemo 30, dakle, ostaju nam 30 i 18. Zatim od 30 oduzimamo 18, pa imamo 12 i 18. Ponavljamo ovaj postupak dok jedan od brojeva ne postane nula. Na kraju, jedini preostali broj (6) je najveći zajednički delilac.

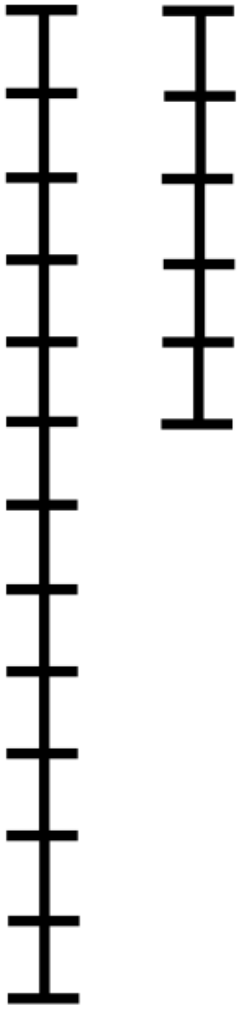
Najveći zajednički delilac (NZD)
Euklidov algoritam

```
def nzd(a, b):
    if a ==0:
       return b
    while b != 0:
        if a > b:
           a = a - b
        else:
           b = b - a
    return a

print(nzd(36,48))
```

```
def nzd2(a, b):
    while b != 0:
       a, b = b, a % b

    return a
```



Najmanji zajednički sadržalac

- Najmanji zajednički sadržalac se može odrediti tako što se prvo odredi NZD:

$$NZS(a,b)*NZD(a,b) = a*b$$

```
def nzs(a, b):
    return a*b/nzd(a,b)
```

Faktorizacija broja

- Dato je n brojeva. Potrebno je faktorisati svaki broj, tj. napisati ga kao proizvod prostih činilaca. Svaki broj faktorisati u formatu

p1^a1 * p2^a2 * ... * pk^ak , gde su p1 ≤ p2 ≤ ... ≤ pk svi prosti činioci datog broja (u rastućem redosledu), a a1, a2, ..., ak - njihovi odgovarajući izložioci. Izmedju brojeva i simbola '*' i '^' ne sme biti razmaka. Takođe, ukoliko je izložilac nekog broja jednak 1, treba ga svejedno ispisati.

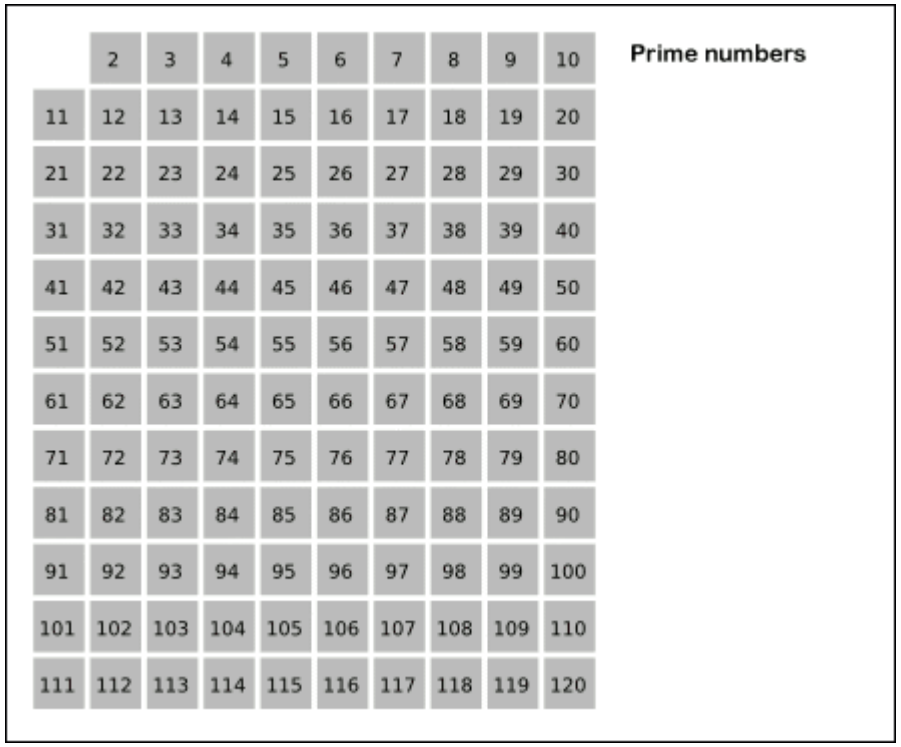
10	2^1*5^1
23	23^1
180	2^2*3^2*5^1

Faktorizacija broja

```
n = int(input())
s = ''
i = 2
while n!=1:
    if n % i==0:
        b = 0
        while n % i==0:
            n //=i
            b +=1
        if s=='':
            s = s + str(i)+'^'+str(b)
        else:
            s = s + '*' + str(i)+'^'+str(b)
    i +=1
print(s)
```

Eratostenovo sito

- Postupak za određivanje prostih brojeva manjih od nekog zadatog broja **n**



Algoritam:

1. Napišite sve brojeve od 2 do n
2. Počevši od prvog broja na spisku (dvojka) precrtajte sa spiska sve brojeve deljive sa dva i upišite da je dvojka prost broj.
3. Ponavljajte postupak sa sledećim neprecrtanim brojem m. Dakle, precrtajte sve brojeve deljive sa m, a njega samog obeležite da je prost.

Eratostenovo sito

```
def nadji_proste_brojeve(n):
    prosti = set()
    slozeni = set()

    for i in range(2, n + 1):
        if i not in slozeni:
            prosti.add(i)
            for j in range(i * i, n + 1, i):
                slozeni.add(j)
    return prosti

print(nadji_proste_brojeve(100))
```

Vrednost polinoma

- Napisati program kojim se za **n** koeficijenata polinoma i realan broj **x**, računa vrednost polinoma.

$$f(x) = a_0 + a_1 x + \cdots + a_n x^n$$

```
def vrednostP(x, polinom):
    result = 0
    for i in range(len(polinom)):
        result += (polinom[i]*x**(len(polinom)-1-i))
    return result

Polinom = [15, 0, -13, 2, -1]
print(vrednostP(2,Polinom))
191
```

$$P(x) = 15x^4 - 13x^2 + 2x - 1$$

Koeficijenti su dati u listi od najveceg ka najmanjem

Hornerova šema

- Ako je $f = a_0 + a_1 x + \cdots + a_n x^n$, tada vrednost

$$f = a_0 + a_1 c + \cdots + a_n c^n$$

se izračunava sa $(1 + 2 + \cdots + n) + n = \frac{n(n+3)}{2}$ operacija + i ·

- Kako sa što manje operacija izračunati $f(c)$?
- Odredimo količnik i ostatak pri deljenju polinoma f sa $x - c$.
- Neka je $f = q(x - c) + f(c)$, gde je $q = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}$

Hornerova šema

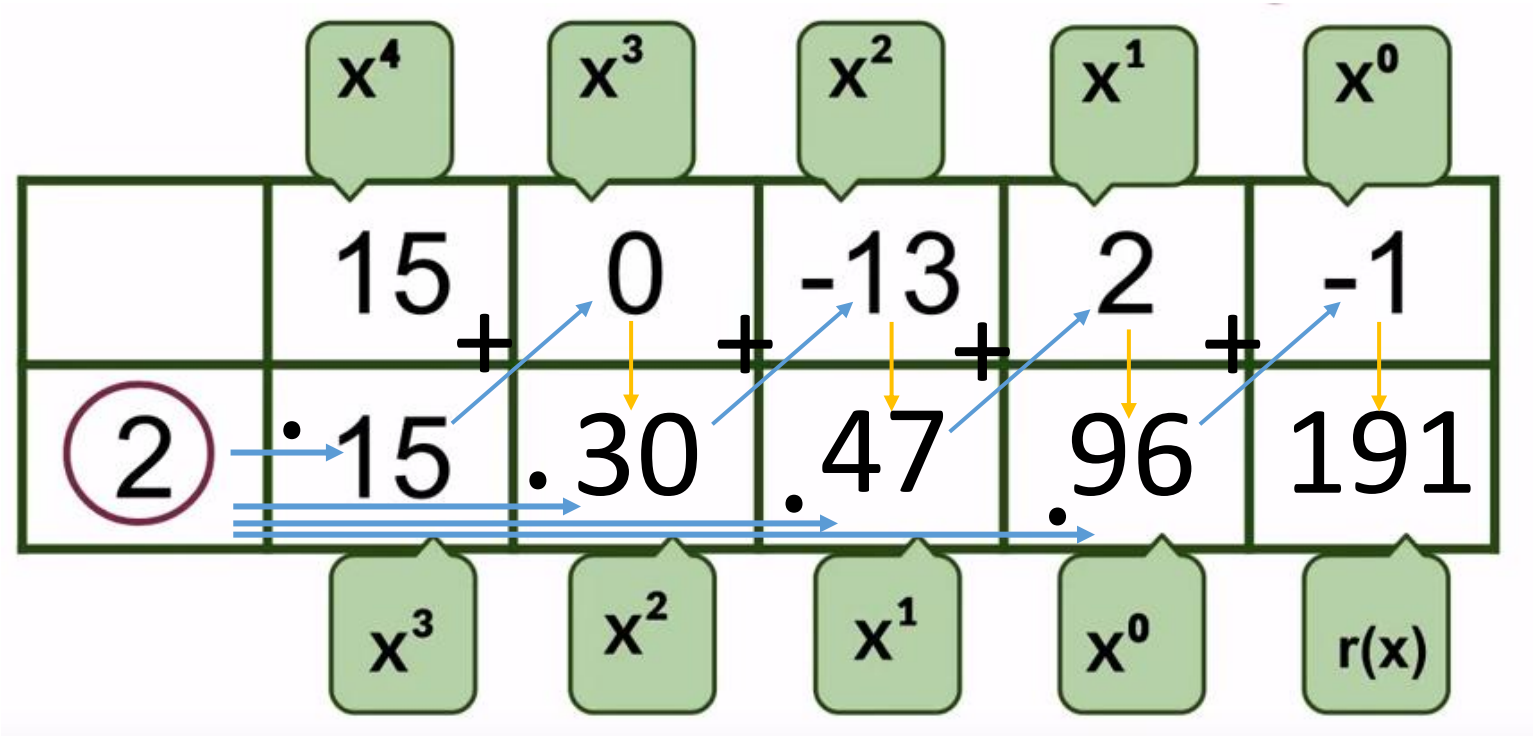
$$a_0 + a_1 c + \cdots + a_n c^n = (b_0 + b_1 x + \cdots + b_{n-1} x^{n-1})(\mathrm{x} - \mathrm{c}) + f(c)$$

Izjednačavanjem koeficijenata dobijamo:

c	a_n	a_{n-1}	$\ldots$	a_1	a_0
	b_{n-1}	b_{n-2}	$\ldots$	b_0	$f(c)$
	II	II	$\ldots$	II	II
	a_n	$a_{n-1}+b_{n-1}c$		a_1+b_1c	a_0+b_0c

Hornerova šema

Primer: $P(x) = 15x^4 - 13x^2 + 2x - 1 = (x - 2) \cdot Q(x) + r$



Hornerova šema

- Napisati program kojim se za **n** koeficijenata polinoma i realan broj **x**, računa vrednost polinoma.

```
def horner(x, polinom):
    result = 0
    for coefficient in polinom:
        result = result * x + coefficient
    return result

Polinom = [15, 0, -13, 2, -1]
print(horner(2,Polinom))
```

Broj dana između dva datuma

- Napisati program kojim se određuje broj dana između dva datuma.

Ulaz	Izlaz
7 4 1953 9 12 2016	23257