PRAKTIKUM IZ PROGRAMIRANJA 1

VEŽBE 10

Ana Vidosavljević, Marija Trnavac

Napisati program koji u nizu od **n** različitih celih brojeva pronalazi maksimalan podskup koji obrazuje seriju susednih brojeva.

Ulaz: Sa standardnog ulaza se unosi broj članova niza **n**, a zatim u narednih **n redova** se unose elementi
niza.

Izlaz: Na standardni izlaz se ispisuje maksimalan podskup niza koji sadrži seriju susednih brojeva.

Primer:

Ulaz: 7
1
4
-4
9
5
2
3

Izlaz: [1, 2, 3, 4, 5]

Primer 1

```
n = int(input())

niz = []

for i in range(n):
    niz.insert(i, int(input()))

niz.sort()
max_serija = []

i = 0
while i < n - 1:
    serija = [niz[i]]

    for j in range(i + 1, n):
        if(niz[j - 1] + 1 == niz[j]):
            serija.append(niz[j])
        else:
            break
    i = j
    if(len(serija) > 1 and len(max_serija) < len(serija)):
        max_serija = serija

if(len(max_serija) > 1):
    print(max_serija)
else:
    print("Nema serije susednih brojeva!")
```

Napisati program koji za dati string **s**, sortira njegove karaktere u nerastućem poretku na osnovu broja ponavljanja karaktera. Ukoliko je broj ponavljanja karaktera jednak, karakteri su uređeni po abecednom redu.

Ulaz: Sa standardnog ulaza se unosi string.

Izlaz: Na standardni izlaz ispisati transformisani string

Primer:

Ulaz: tree Izlaz: eert

```
def sortiraj(slova):
    for i in range(26-1):
        for j in range(i+1, 26):
            if (slova[i][1] < slova[j][1]) or
(slova[i][1] == slova[j][1] and slova[i][0] >
slova[j][0]):
                pom = slova[i]
                slova[i] = slova[j]
                slova[j] = pom
```

Primer 2

```
def formiraj(rec):
    slova = [0 for _ in range(26)]

    for i in range(26):
        slova[i] = [chr(97 + i), 0]

    for c in rec:
       slova[ord(c) - 97][1] += 1

    return slova

rec = input("Unesite rec: ")
slova = formiraj(rec)
sortiraj(slova)
nova_rec = ""

for i in range(26):
   if (slova[i][1] != 0):
      nova_rec += slova[i][0]*slova[i][1]

print(nova_rec)
```

Matricom reda **N** data je tabela jesenjeg dela fudbalskog šampionata, čiji su elementi:

$$a[i][j]=\begin{cases} 0, & ako\ je\ ekipa\ \boldsymbol{i}\ izgubila\ od\ ekipe\ \boldsymbol{j} \\ 1, & nerešen\ rezultat \\ 2, & ako\ je\ ekipa\ \boldsymbol{i}\ pobedila\ ekipu\ \boldsymbol{j} \end{cases}$$

Napisati program kojim se izračunava:
a) broj ekipa koje su imale više pobeda nego poraza
b) broj ekipa koje su prošle prvenstvo bez poraza

Sadržaj na glavnoj dijagonali zanemariti.

```
def visePobeda(a, n):
    brojEkipa = 0
    for i in range(n):
        brojPobeda = 0
        brojPoraza = 0
        for j in range(n):
            if (i != j):
                if (a[i][j] == 2):
                    brojPobeda += 1
                elif (a[i][j] == 0):
                    brojPoraza += 1
        if (brojPobeda > brojPoraza):
            brojEkipa += 1

    return brojEkipa
```

```
def bezPoraza(a, n):
    brojEkipa = 0
    for i in range(n):
        bezPoraza = True
        for j in range(n):
            if (i != j):
                if (a[i][j] == 0):
                    bezPoraza = False
                    break
        if (bezPoraza):
            brojEkipa += 1

    return brojEkipa
```

```
n = int(input("Unesi n: "))
a = []

for i in range(n):
    a.append(list(map(int, input().split())))

print("Broj ekipa koje su imale više pobeda nego poraza:",
visePobeda(a, n))
print("Broj ekipa koje su prošle prvenstvno bez poraza:",
bezPoraza(a, n))
```

Na papiru podeljenom na **n×n** jediničnih kvadratića, nacrtano je nekoliko pravougaonika (paralelno ivicama papira) koji se ne dodiruju, osim eventualno jednim temenom. Napisati program koji određuje broj pravougaonika na papiru.

Ulaz: U prvom redu standardnog ulaza data je dimenzija table **n** ≤ 10, a zatim je u narednim redovima zadata matrica koja sadrži jedinice i nule, tako da su jedinice upisane na mestima na kojima su nacrtani pravougaonici.

Izlaz: Na standardni izlaz ispisati traženi broj pravougaonika.

Primer:

Ulaz	***Izlaz***
7 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1	4

```
def ucitaj():
    n = int(input())
    A = []
    for v in range(n):
        A.append(list(map(int, input().split())))
    return (A, n)

(A, n) = ucitaj()
broj_pravougaonika = 0

for v in range(n):
    for k in range(n):
        # ako je A[v][k] gornje levo teme pravougaonika
        if (A[v][k] == 1 and (v == 0 or A[v-1][k] != 1) and
        (k == 0 or A[v][k-1] != 1)):
            broj_pravougaonika += 1

print(broj_pravougaonika)
```

Napisati program kojim se u jednoj liniji unosi korektno zapisan prefiksni izraz, a zatim određuje njegova vrednost. Operandi u izrazu su prirodni brojevi razdvojeni blanko simbolom, a operacije su iz skupa +, -, *, / pri čemu je / celobrojno deljenje.

Primer: Prefiksnom izrazu **+ 31 4** odgovara infiksni izraz **31+4**, a prefiksnom izrazu **/ - 30 * 12 7 34** odgovara **(30 - 12 * 7)/34**.

```
def sab(niz, oper):
    return int(niz[oper + 1]) + int(niz[oper + 2])

def oduz(niz, oper):
    return int(niz[oper + 1]) - int(niz[oper + 2])

def mnoz(niz, oper):
    return int(niz[oper + 1]) * int(niz[oper + 2])

def delj(niz, oper):
    return int(niz[oper + 1]) // int(niz[oper + 2])

izraz = str(input())
n = len(izraz)
r = 0
niz = []
niz = izraz.split(" ")
oper = 1
```

Primer 5

```
while(oper != -1):
    oper = - 1
    for i in range(len(niz)):
        if niz[i] == "+" or niz[i] == "-" or niz[i] == "*" or niz[i] == "/":
            oper = i

    if oper == -1:
        break
    if niz[oper] == "+":
        r = sab(niz, oper)
    if niz[oper] == "-":
        r = oduz(niz, oper)
    if niz[oper] == "*":
        r = mnoz(niz, oper)
    if niz[oper] == "/":
        r = delj(niz, oper)

    niz.pop(oper)
    niz.pop(oper)
    niz.pop(oper)
    niz.append(1)

    for i in range(len(niz) - 1, oper, -1):
        niz[i] = niz[i - 1]
    niz[oper] = r

print(int(niz[0]))
```