PRAKTIKUM IZ PROGRAMIRANJA 1
VEŽBE 9

Milica Vasović, Jana Brzaković, Ana Vidosavljević

Zadata je reč sastavljena od malih slova rimske abecede (koja je podskup engleske). Potrebno je šifrovati, odnosno dešifrovati poruku. Cezarova šifra je tip šifre zamenjivanja u kome se svako slovo teksta koji se šifruje menja slovom abecede koje se dobije pomeranjem polaznog slova za određeni broj mesta, ciklično po abecedi. Na primer, ako se vrši pomeranje za tri mesta, slovo **a** bi se šifrovalo slovom **d**, **b** slovom **e**, …, dok bi se slovo **w** šifrovalo slovom **z**, slovo **x** slovom **a**, slovo **y** slovom **b** i slovo **z** slovom **c**. Na primer, reč **papagaj** bi se šifrovala sa **sdsdjdm**.

Ulaz: U prvoj liniji standardnog ulaza nalazi se reč ne duža od 100 slova, u drugoj liniji se nalazi ceo broj **N** koji predstavlja pomak (1 <= N < 26), a u trećoj ceo broj **S** koji predstavlja smer šifrovanja. Ako je **S** = 1 potrebno je **šifrovati**, a ako je **S** = 2 potrebno je **dešifrovati** reč.

Primer 1

```
rec = input()
pomeraj = int(input())
smer = int(input())
rez_slova = []

if smer == 1: # sifrovanje
    for c in rec:
        cn = ord(c) + pomeraj

        if cn > ord('z'):
            cn -= 26

        rez_slova.append(chr(cn))

else: # desifrovanje
    for c in rec:
        cn = ord(c) - pomeraj

        if cn < ord('a'):
            cn += 26

        rez_slova.append(chr(cn))
```

PRIMER 1

```
nova_rec = ""

for slovo in rez_slova:
    nova_rec = nova_rec + slovo

print(nova_rec)
```

U datom nizu prirodnih brojeva dužine n, odredi broj grupa od dva ili tri elementa takvih da je suma svih elemenata grupe deljiva sa 3. Na primer, u nizu [2, 1, 3, 10] imamo 4 takve grupe: [2, 1], [2, 10], [2, 1, 3], [2, 3, 10].

PRIMER 2

```
n = int(input("Unesi n: "))

niz = []

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

br = 0

for i in range(n):
    for j in range(i + 1, n):
        if((niz[i] + niz[j]) % 3 == 0):
            br += 1
        for k in range(j + 1, n):
            if((niz[i] + niz[j] + niz[k]) % 3 == 0):
                br += 1
print(br)
```

Đaci uvežbavaju sabiranje trocifrenih brojeva. Učiteljica diktira brojeve redom cifru po cifru. Napiši program koji na osnovu učitanih cifara izračunava i ispisuje traženi zbir.

Ulaz: U šest linija standardnog ulaza zadato je šest cifara.

Izlaz: Na standardni izlaz ispisati jedan ceo broj - traženi zbir.

Primer

Ulaz	**Izlaz**
1 2 3 4 5 6	579

PRIMER 3

```
a = [int(input()) for i in range(3)]
a.reverse()
b = [int(input()) for i in range(3)]
b.reverse()

z = [0, 0, 0, 0]
prenos = 0
for i in range(3):
    zbirCifara = a[i] + b[i] + prenos
    z[i] = zbirCifara % 10
    prenos = zbirCifara // 10
z[3] = prenos

j = 3
while j > 0 and z[j] == 0:
    j -= 1
while j >= 0:
    print(z[j], end="")
    j -= 1
print()
```

Napisati program koji proverava da li je sudoku rešenje validno. Sudoku je rešen ako:

- Svaki red mora sadržati cifre 1-9 bez ponavljanja.
- Svaka kolona mora sadržati cifre 1-9 bez ponavljanja.
- Svaka od devet 3 x 3 podmatrica mora sadržati cifre 1-9 bez ponavljanja.

Napisati program koji proverava da li je sudoku rešenje validno. Sudoku je rešen ako:

- Svaki red mora sadržati cifre 1-9 bez ponavljanja.
- Svaka kolona mora sadržati cifre 1-9 bez ponavljanja.
- Svaka od devet 3 x 3 podmatrica mora sadržati cifre 1-9 bez ponavljanja.

```
sudoku = [
    [5, 3, 4, 6, 7, 8, 9, 1, 2],
    [6, 7, 2, 1, 9, 5, 3, 4, 8],
    [1, 9, 8, 3, 4, 2, 5, 6, 7],
    [8, 5, 9, 7, 6, 1, 4, 2, 3],
    [4, 2, 6, 8, 5, 3, 7, 9, 1],
    [7, 1, 3, 9, 2, 4, 8, 5, 6],
    [9, 6, 1, 5, 3, 7, 2, 8, 4],
    [2, 8, 7, 4, 1, 9, 6, 3, 5],
    [3, 4, 5, 2, 8, 6, 1, 7, 9]
]
```

Resenje je tacno

```
sudoku = [
    [5, 3, 4, 6, 7, 8, 9, 1, 2],
    [6, 7, 2, 1, 9, 5, 3, 4, 8],
    [1, 9, 8, 3, 4, 2, 5, 6, 7],
    [8, 5, 9, 7, 6, 1, 4, 2, 3],
    [4, 2, 6, 8, 5, 3, 7, 9, 1],
    [7, 1, 3, 9, 2, 4, 8, 5, 6],
    [9, 6, 1, 5, 3, 7, 2, 8, 4],
    [2, 8, 7, 4, 1, 9, 6, 3, 5],
    [3, 4, 5, 2, 8, 6, 1, 9, 9]
]
```

Resenje nije tacno

```
def validan_sudoku(mat):
    for i in range(9):
        if not validna_vrsta(mat, i) or not validna_kolona(mat,
    i):
            return False

    for i in range(0, 9, 3):
        for j in range(0, 9, 3):
            if not validna_podmatrica(mat, i, j):
                return False

    return True

def validna_kolona(mat, kolona):
    procitano = []
    for i in range(9):
        br = mat[i][kolona]
        if procitano.count(br)>=1:
            return False
        procitano.append(br)
    return True
```

```
def validna_vrsta(mat, vrsta):
    procitano = []
    for br in mat[vrsta]:
        if procitano.count(br)>=1:
            return False
        procitano.append(br)
    return True

def validna_podmatrica(mat, poc_vrsta, poc_kolona):
    procitano = []
    for i in range(3):
        for j in range(3):
            br = mat[poc_vrsta + i][poc_kolona + j]
            if procitano.count(br)>=1:
                return False
            procitano.append(br)
    return True
```

PRIMER 4

```
sudoku = [
    [5, 3, 4, 6, 7, 8, 9, 1, 2],
    [6, 7, 2, 1, 9, 5, 3, 4, 8],
    [1, 9, 8, 3, 4, 2, 5, 6, 7],
    [8, 5, 9, 7, 6, 1, 4, 2, 3],
    [4, 2, 6, 8, 5, 3, 7, 9, 1],
    [7, 1, 3, 9, 2, 4, 8, 5, 6],
    [9, 6, 1, 5, 3, 7, 2, 8, 4],
    [2, 8, 7, 4, 1, 9, 6, 3, 5],
    [3, 4, 5, 2, 8, 6, 1, 7, 9]
]

if validan_sudoku(sudoku):
    print("Resenje je tacno")
else:
    print("Resenje nije tacno")
```

Napisati program koji za uneti string s i ceo broj k, smanjuje string izbacivanjem k uzastopnih ponavljanja istog slova sve dok je to moguće. Na kraju, odštampati smanjeni string.

Primer 1:

Ulaz:
“qdfffdd”
3

Izlaz:
“q”

Primer 2:

Ulaz:
“pippip”
2

Izlaz:
“”

```
def izbaci_k_uzastionih(k, s):
    if k == 1:
        return ""

    while True:
        promenjen = False
        i = 0
        while i < len(s):
            ponavljanje = s[i] * k
            pozicija = s.find(ponavljanje, i)
            if pozicija != -1:
                s = s[:pozicija] + s[pozicija + k:]
                promenjen = True
                break
            else:
                i += 1
        if not promenjen:
            break

    return s
```

PRIMER 5

```
s = input()
k = int(input())

print(izbaci_k_uzastopnih(k, s))
```