# Tutoriat 2 – Șiruri de caractere
## - rezolvări -

1. Se citesc un șir de caractere `s` și două subșiruri ale sale `p` și `q`. Interschimbați prima apariție a subșirului `p` în șirul `s` cu ultima apariție a subșirului `q` în șirul `s`. Exemplu: `s` = “pepene pe pene”, `p` = “pe”, `q` = “ne” => “nepene pe pepe”.

```
#V1
s = input()
p = input()
q = input()

s = s.replace(p, q, 1)
s = s[::-1].replace(q[::-1], p[::-1], 1)[::-1]
print(s)

#V2
s = input()
p = input()
q = input()
poz = s.find(p)
s = s[:poz] + q + s[poz + len(p):]
poz = s.rfind(q)
s = s[:poz] + p + s[poz + len(q):]
print(s)
```

2. Se dă un șir de caractere reprezentând un cuvânt. Să se afișeze lista prefixelor și lista sufixelor cuvântului dat.

```
#V1
s=input("SIR: ")
for i in range(0, len(s)):
    print(s[:len(s)-i])
for i in range(0, len(s)):
    print(s[i:])

#V2
s = input()
prefixe = [ s[:i] for i in range( len(s), 0, -1 ) ]
print( *prefixe, sep="\n" )
sufixe = [ s[i:] for i in range( len(s) ) ]
print( "\n".join(sufixe) )
```

3. Se citește un șir de caractere. Să se înlocuiască toate vocalele cu simbolul ’**#**’. Exemplu: ”Ana are mere” => ”#n# #r# m#r#”.

```
#V1
s = input()
for i in "aeiouAEIOU":
    s = s.replace(i, "#")
print(s)

#V2
s = input()
voc = "aeiouAEIOU"
ls = ['#' if i in voc else i for i in s]
print(''.join(ls))
```

4. Se citește un șir de caractere. Afișați șirul format prin eliminarea caracterelor de pe poziții impare. Exemplu: ”abcdef” => ”ace”.

```
s = input()
print( s[::2] )
# print( s[1::2] ) - eliminam pozitiile pare
```

5. Se dă de la tastatură o formulă ce conține două numere în binar (nu se garantează că numerele au același număr de biți) și operatorul & (conjuncția binară). Afișați rezultatul operației (tot în binar). Exemplu: “1011010&1011” => “1010”.

```
s = input().split("&")
a = s[0]
b = s[1]
if len(a) > len(b):
    a = a[len(a) - len(b):]
elif len(b) > len(a):
    b = b[len(b) - len(a):]
rez = ""
for i in range( len(a) ):
    rez += str( int(a[i]) & int(b[i]) ) # reprezentarea binara a lui 0 = 0, iar a lui 1 = 1, deci putem aplica direct &
    # rez += str( int(a[i]) * int(b[i]) ) - merge si cu produsul - daca avem un bit 0, rezultatul e 0
print(rez)
```

6. Se citesc două cuvinte, conținând litere mari și mici ale alfabetului englez. Considerând că minuscula și majuscula reprezintă același simbol (de exemplu, 'a' = 'A'), verificați dacă cele două cuvinte sunt anagrame. Două cuvinte sunt anagrame, dacă unul dintre cuvinte poate fi format prin schimbarea ordinii literelor celuilalt. Exemplu: “rutina” și “unitar” sunt anagrame.

```
s1 = input()
s1 = s1.lower()
s2 = input()
s2 = s2.lower()
frecv = [0] * ( ord('z') - ord('a') )
for litera in s1:
    frecv[ ord( litera ) - ord('a') ] += 1
for litera in s2:
    frecv[ ord(litera) - ord('a') ] -= 1
print( frecv == [0] * ( ord('z') - ord('a') ) )
```

7. Se citește un șir de caractere. Determinați cuvântul palindrom de lungime maximă. Exemplu: ”121 prieteni analizeaza Un capac aeriSIrea  radar” => “aeriSIrea”.

```
s = input().split()
max = ""
lmax = 0
for cuv in s:
    cop = cuv.lower()
    if cop == cop[::-1] and len( cop ) > lmax:
        lmax = len( cop )
        max = cuv
print( max )
```