Programare Procedurală
- Laborator 3 –

1. Stiind că `l = [1, 2, 3, 4, 5, 6, 7, 8, 9]`, care este outputul urmatoarelor comenzi:

```
a) print(l[7:5])
b) print(l[0:1000])
c) print(l[-4:-1])
d) print(l[-4:0])
e) print(l[-4:0:-1])
f) print(l[-4:])
g) print(l[-1:-4])
h) print(l[::])
i) print(l[::100])
j) print(l[::-100])
```

Răspundeți, justificați și apoi rulați pentru a vă verifica răspunsurile.

2. Stiind că `l = [1, 2, 3]`, care este outputul următorului fragment de cod, rulat în ordinea aceasta:

```
In[1]: l[len(l):] = [4, 5, 6]
       print(l)
In[2]: l[:0] = [-3, -2, -1, 0]
       print(l)
```

Răspundeți, justificați și apoi rulați pentru a vă verifica răspunsurile.

3. Fie `l` o listă oarecare. Completați codul `l[_:_:-1]` în locurile marcate cu "_" cu expresii (nevide) potrivite astfel încât rezultatul rulării acestuia să fie identic cu cel al rulării codului `l[::-1]`.

4. Care este outputul următoarelor comenzi:

a) `print([1, 2, 3] * 1)`
b) `print([1, 2, 3] * 0)`
c) `print([1, 2, 3] * -1)`

Răspundeți, justificați și apoi rulați pentru a vă verifica răspunsurile.

5. Care dintre codurile următoare au ca efect inserarea jumătății numărului după orice număr par dintr-o listă l, fără a folosi liste auxiliare? Justificați, scrieți ce se va afișa pentru următoarele liste l = [1, 14, 18, 23], l = [1, 14, 18, 23, 6, 24], l = [1, 24, 18, 23, 12] și apoi rulați pentru a vă verifica răspunsurile.

a)

```
for i in range(len(l)):
    if l[i] % 2 == 0:
        l.insert(i+1, l[i] // 2)
        i = i + 1
    i = i + 1
print(l)
```

b)

```
i = 0
while i < len(l):
    if l[i] % 2 == 0:
        l.insert(i+1, l[i] // 2)
    i = i + 1
print(l)
```

c)

```
i = 0
while i < len(l):
    if l[i] % 2 == 0:
        l.insert(i+1, l[i] // 2)
        i = i + 1
    i = i + 1
print(l)
```

d)

```
for el in list(l):
     if el % 2 == 0:
          i = l.index(el)
         l.insert(i+1, l[i] // 2)
print(l)
```

6. Fie lista **l_1 = [1, 2, 240, 120, 2, 3, 18, 12, 22, 28, 1004].** Să se șteargă:
   a. toate valorile pare;
   b. toate valorile prime.

7. Se citește **n**, apoi **n** numere. Să se rearanjeze numerele și să se memoreze într-o listă astfel încât toate valorile nule să fie la finalul acesteia.

8. Care este efectul execuției secvenței de cod?

```
import math
L = []
x = int(input())
for n in range(2, x + 1):
    for factor in L:
        if n % factor == 0 and factor <= math.sqrt(x):
            break
    else:
        L.append(n)
print(L)
```

9. Fie listele **l_1, l_2** și **n, m, k<=n*m** numere naturale, citite. Se citesc **n, m** valori întregi ce se memorează în lista **l_1**, respectiv **l_2**. Să se afișeze primele **k** perechi **(l1, l2), l1∈l_1, l2∈l_2**, ce au suma minimă.

10. Fie o listă ce conține liste și tupluri încuibate (ex.: l = [1, [2,3], [4, 5, (6, (7, 8, 9))], 10]. Să se transforme într-o listă simplă, fără a folosi liste suplimentare:

a) ignorând ordinea elementelor;
b) păstrând ordinea elementelor.