6. a)

**Argumentare complexitate: ()**

**(i)**Dimensiunea problemei este: **(n)**

In Python folosim functia len() pentru conditia de oprire, dar vom nota aseasta cu “n” ca fiind numarul de elemente ale listei.

Dimesniunea problemei este “n”, deoarece acesta este o cracteristica a datelor de intrare, intrare ce ne influenteaza pasii pe care ii avem de parcurs.

**(ii)**Operatia dominanta este: **sum+= x[j]**

Aceasta este operatia cea mai din interior si este cea mai specifica operatie pentru media aritmetica din cele prezente in algoritm.

* **(iii)**Numarul de executii ale operatiei dominante:

Numarul de executii ale operatiei dominante este dat de numarul de executii ale celui de-al doilea for. Iar numarul de executii ale celui de-al doilea for e dat de numarul de executii ale primului for din care scadem 1.

for i in range(len(x)): n+1  
 sum = 0  
 for j in range(i+1): i+1  
 sum += x[j]

**Primul for:**

for i in range(len(x)):

Se merge de la 0 la n, adica de “n-0 + 1” ori = **n+1 (se executa de n+1 ori)**

* al doilea for se executa de n+1-1 ori, adica de n ori.
* Numarul de executii ale operatiei dominanate este: (care se poate rezolva cu suma Gauss)

**(iV)**Complexitatea este data de cele 2 for-uri imbricate

for i in range(len(x)):   
 sum = 0  
 for j in range(i+1):  
 sum += x[j]

**Primul for:**

for i in range(len(x)):

Se merge de la 0 la n, adica de “n-0 + 1” ori = **n+1 => (se executa de n+1 ori)**

**Al doilea for:**

for j in range(i+1):

Face parte din corpul primului for  **=>** (**se executa de n ori)**

**[CONCLUZIE]**

Prin urmare, parcurgerea tabloului se face **de (n+1)\*n** ori, adica de **+ n** ori.

Dependenta fata de dimenisunea problemei este patratica.

Pentru ca nu avem cum sa iesim alcumva din repetitive, complexitatea este **() (Teta).**