Assignment 1 - Theoretical part

Question 1

```
Algorithm ComputeAverage(A,n)
   sum ← 0
   for i = 0 to n-1 do
      sum ← sum + A [i]
   end
return  sum / n
```

Question 2&3

a- Number of primitive operations T(n) of Algorithm Compute(n), counting assignments, comparisons, and returns only

Assignments = 2 + 2n
Coparisons = n +1
Return = 1
So T(n) = 3n + 4

b- In pseudo code, describe Algorithm RecursiveCompute(n), an algorithm that outputs the sum of all integers from 1 to n as recursive algorithm.

```
Algorithm RecursiveCompute(n)
   if n = 1 then
      return 1
   else
      return  n + RecursiveCompute  (n-1)
   end
```

c - Number of primitive operations of Algorithm RecursiveCompute(n), counting assignments, comparisons, and returns only

$$T(n) = \begin{cases} 2 \text{ if } n = 1 \\ T(n-1) + 3 \text{ if } n \geq 2 \end{cases}$$

So the number of primitive operations is 3n -1

d - In pseudo code, describe Algorithm ComputeFast(n), an algorithm that outputs the sum of all integers from 1 to n in constant time

```
Algorithm ComputeFast(n)
sum ← n * ( n +1 ) / 2
return sum
```

Question 4

- a) Order the following functions by their growth rate (big-Oh)

f_7 ,f_0 ,f_1 ,f_4 ,f_6 ,f_2 ,f_5 ,f_3

b) Functions which are big-theta of one another

g_0 and g_5 ; g_1 and g_7 ; g_2 and g_3 .

g_4 and g_6 are not big-theta of one another. And neither of them are big- theta of the above functions.

Question 5

Best case

T(n) = 4 if there is at least 1 element in the array. If there is no element in the array T(n) = 3

Worst case : T(n) = 3n + 3