**Data Structures & Algorithms – Problem set 1**

**Submitted by Itay Shapira ID 034250043**

Question 1:

Base step: n=1 → , which is dividable by 9.

Inductive step: Let’s assume is true for n=k, Hence is true

Let’s prove for n=k+1:

* => =>
* The first part assumed to be dividable by 9, let’s prove by second induction the second part is also dividable by 9.
* Base step: n=1 → , which is dividable by 9.
* Inductive step: Let’s assume is true for n=k, Hence is true
* Let’s prove for n=k+1:
* , both part are dividable by 9.
* Hence, is dividable by 9.

Question 2:

Let’s prove: .

Let’s solve left side first:

, for all n>0, hence.

Now right side:

, true when

.

Hence =1/6 for and for all

Question 3:

The order of the functions (ascending order) :

1. O(1)
2. F5 =
3. F4 = n/10 =
4. = O( for n
5. F3 =

Proof for the pairs:

:

which is a constant.

:

which is a constant.



For every n and = F7= = \*F5

1. **F5 = O(F4):**

Proof by induction:

Claim: For all n≥1, log(n)≤n.

Base: for n=1, since 0<1.

Now suppose n≥1 and log(n)≤n. Then we need to prove log(n+1)≤n+1. Because n≥1 we can assume that log(n+1)≤log(2n) = log(n)+1≤n+1 by the inductive hypothesis.

1. **F4 = 0(F6):**

Because for c for every n: n/10 ≤ n ≤ c\*

1. **F3 = 0(F6):**

= for any n as log(n) is monotone, thus F3

Question 4:

The definition of Big O Notation is based on the fact that there exist two **constants** C, that holds:

for any N>. These two cannot be changed and should hold for every value.

In the induction attempt in the question the assumption claimed that there is a constant that holds the proof. But in the proof step in order to verify the assumption C was changed which undermine the whole proof. C was needed to be changed in order for the proof to hold which directly contradicts the definition the induction and bigO.

Question 5:

A.

This means,

let’s prove:

**Let’s solve the right side:**

If

If

Thus

**For the left side:**

Reversing the role of F(n) and G(n) also show that

which enough to prove that log(f(n)) = Θ(log(g(n)))

Thus the statement is true.

B.

Let for example f(n)=2n and g(n)=n => .

Let’s test it with the limit definition of :

, thus the statement is false.

Question 6:

Function findPair(L,k): {

NodeA = L.First

For I= 1 to length L -1:

For J=i+1 to length L:

NodeB = NodeA.next

If NodeA.value + NodeB.value = K return True.

NodeB = NodeB.next

NodeA =NodeA.next

Return False

}

In the best case scenario the first two items will justify the condition which will account as 1 comparison = O(1). In the worst case scenario none of pair items will validate the requirements, in this case the program will run through the whole list and for each item on the list it will ran again on the rest of the list to try to find an item that together will justify the condition, thus it will run n-1 of the first run, then n-2, n-3, till 1 -> n-1 times. The sum of this arithmetic sequence is = O()