National University of Computer & Emerging Sciences, Karachi
Fall-2018 Department of Computer Science
Mid Term - I
1st October 2018, 11:00 am– 12:00 noon

Course Code: CS302	**Course Name:** Design and Analysis of Algorithm
Instructor Name / Names: Dr. Muhammad Atif Tahir, Subhash Sagar and Zeshan Khan	
Student Roll No:	**Section:**

Instructions:

- Return the question paper.
- Read each question completely before answering it. There are **7 questions** on **2 pages.**
- In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper.

Time: 60 minutes. **Max Marks: 12.5**

Question # 1 **[2 marks]**

Use worst case analysis to construct a function T(n) that gives the runtime complexity of the algorithm as a function of n. Simple example is shown in **Figure: 1.**

```
Function_A(a, n) {
        int i, j, temp, flag=true;
        for (i=0; i<n-1;i++) {
                for (j=0; j<n-i-1; j++) {
                        if (a [j] > a [j+1]) {
                                temp = a[j];
                                a [j] = a [j+1];
                                a [j+1] = temp ;
                        }
                }
        }
}
```

Figure: 1

```
Example:
Int a = 0;------------ c1
Int n=10;------------ c2
For (int i=0;i<n, i++){ ---- c3* (n+1)
        a=a+1;-------- c4* n
}
Total Cost = T (n) =
c1+c2+c3(n+1)+c4n
T (n)= (c3+c4)n+(c1+c2+c3)
T(n) = O(n)
```

Solution # 1

```
Function_A(a, n) {
        int i, j, temp, flag=true;                          c1          } (0.5)
        for (i=0; i<n-1;i++) {                              c2*n        }
                for (j=0; j<n-i-1; j++) {                   c3*n(n+1)/2 ]
                        if (a [j] > a [j+1]) {              c4*n(n+1)/2 ]
                                temp = a[j];                c5*n(n+1)/2 ] (1.0)
                                a [j] = a [j+1];            c6*n(n+1)/2 ]
                                a [j+1] = temp ;            c7*n(n+1)/2 ]
                        }
                }
        }
}
```

Hence T(n) = c1+c2*n+(c3+c4+c5+c6+c7)/2*n(n+1)
= c1+ (c2+(c3+c4+c5+c6+c7)/2)*n+(c3+c4+c5+c6+c7)/2*n^2 $\approx an^2 + bn + d$
$\boldsymbol{T(n) = \theta(n^2)}$ **------- (0.5)**

Question # 2 **[1 mark]**

Let A, B and C are three different algorithms designed for some task $\boldsymbol{T}$. Their worst time complexity are respectively: $f_1(n) = 3n^{20} logn$, $f_2(n) = 2n^{22}$ and $f_3(n) = 90n^{17} + n^{20}$ respectively. Which algorithm is suitable for task T (Explain Briefly?).

Solution # 2

f_3(n) due to lowest growth function → **(1 mark)**

Question # 3 **[0.75*4=3 marks]**

Mark each of following expression by **True** or **False**. State the reason.

a) $\boldsymbol{2^n + n! \in O(n!)}$
b) $\boldsymbol{\frac{n(n+1)}{2} \in \Omega(n)}$
c) $\boldsymbol{\sqrt{10n^2 + 7n + 3} \in \theta(n^2)}$
d) $\boldsymbol{4^{\log_2 n} \in o(n)}$

Solution # 3 **True or False → (0.25) and Correct Reasoning → (0.5)**

True: [As $n!$ is greater than 2^n]
True: [$f(n >= cg(n)$ $satifies\ if\ c >= 1\ and\ n >= 1$],
False: [If we solve square root, equation will be linear so $\theta(n^2)$ is not possible],
False: [n^2 is always greater than n so $o(n)$ is not possible]

Question # 4 **[0.5 marks]**

Find the recursive relation (e.g. T(n)=T(n-1)+n) for the following algorithm:

```
Function_A(n){
        if (n > =1) {
            m = 1;
            return 0;
        }
        else
            m=n/2;
        for (i=1; i<=2; i++)
            Function_B(m);
}
Function_B(n) {
        Function_A (n); }
```

Solution # 4

T(n)=2T(n/2)+1

Question # 5 **[2.5 marks]**

Compute the time complexity of the following recursive relation by using **Recurrence-Tree Method** or **Iterative Substitution Method**.

$\boldsymbol{T(n) = 16T(n/2) + n, T(1) = 1}$

Solution # 5

Q:4 $T(n) = 16T(n/2) + n$

using Iteration

$T(n) = 16T(n/2) + n$

$\Rightarrow 16\left[16T(n/4) + n/2\right] + n$

$= 16^2 T(n/4) + 8n + n$

$= 16^2\left[16T(n/8) + n/4\right] + n + 8n$

$= 16^3 T(n/8) + 64n + 8n + n$

$\vdots$

$= 16^K T(n/2^K) + \underbrace{[n + 8n + 64n + \cdots]}_{K \text{ times}}$

$T(n/2) = 16T(n/4) + n/2$

$T(n/4) = 16T(n/8) + n/4$

For $\frac{n}{2^K} = 1 \Rightarrow K = \log_2 n$

$= 16^K \times T(1) + 8^K n.$

$= 16^{\log_2 n} \times T(1) + 8^{\log_2 n} n.$

$= n^4 + n^4 \Rightarrow n^4$

$\boxed{T(n) = O(n^3)}$

Right Hand Side Eq. (16^k) → (1 mark)

Left Hand Side Eq. (n+8n_+..) → (1 mark)

Final Answer -→ (0.5 marks)

Question # 6

[0.5*4=2 marks]

Solve the following recurrences using **Master's Method.** Give argument, if the recurrence cannot be solved using Master's Method.

a) $T(n) = 9T(n/3) + n$

b) $T(n) = 2^n\, T\left(\frac{n}{2}\right) + n^{n-1}$

c) $T(n) = \sqrt{2}T(n/2) + \log_2 2$

d) $T(n) = 3T(n/3) + n$

Solution # 6 **(0.5 marks) each**

(a) a = 9, b = 3, d = 1. Since a > b^d , Thus T(n) = Theta($n^{\log_3 9}$)

(b) Cannot be applied since f(n) is not polynomial

(c) a =sqrt(20, b= 2, d = 0 (Since log2 = 1 i.e. n^0). Since a > b^d Thus T(n) = Theta($n^{\log_2 \mathrm{sqrt}(2)}$)

(d) a= 3, b = 3, d = 1, a = b^d, Thus T(n) = Theta(nlogn)

Question # 7 **[1.5 marks]**

Consider the following algorithm with $O(n^3)$ complexity. Provide a $O(n^2)$ solution for this algorithm.

```
Algorithm (Matrix A, Matrix B){
        a=n/2
        for (i=0;i<n;i++){
                for (j=0;j<=a;j++){
                        if (j%i==0){
                            j+=1;}
                        for (k=0;k<n;k++){
                                X[i][k]=A[i][k]+B[i][k]
                        }
                }
        }}
```

Solution # 7

```
Algorithm(Matrix A, Matrix B){                                or
 a=n/2
 for(i=0;i<n;i++){
    for(j=0;j<=a;j++){
      if(j%i==0)
        j+=1;
    }
 }
 for(i=0;i<n;i++){
    for(k=0;k<n;k++){
        X[i][k]=A[i][k]+B[i][k];
```

(0.75) (first block)

(0.75) (second block)

```
    }}}
Or
  for(i=0;i<n;i++){
    for(k=0;k<n;k++){
                X[i][k]=A[i][k]+B[i][k];
    }}
```

$$\text{(1.5)}$$