National University of Computer & Emerging Sciences, Karachi
Fall-2019 Department of Computer Science
Mid Term-1
24th September 2018, 11:00 AM – 12:00 PM

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

Instructions:
- Return the question paper.
- Read each question completely before answering it. There are **5 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 **[1.5 marks]**

Are these following statement true or false? Prove your answer by computing the values of n_0, c_1, c_2 or by contradiction. [Θ is Theta]

$\boldsymbol{n^2 + 4^5 = \Theta(n^2)}$
True
$c_1 n^2 \leq n^2 + 4^5 \leq c_2 n^2$
$c_1 \leq 1 + 4^5/n^2 \leq c_2$
$for\ n_0 = 1$
$c_1 \leq 1 + 1024 \leq c_2$
$c_1 \leq 1025 \leq c_2$
$for\ n = \infty$
$c_1 \leq 1 + 0 \leq c_2$
$c_1 \leq 1 \leq c_2$

$\boldsymbol{2^n + 2n = \Omega(n^2)}$
True
$2^n + 2n \geq c_1 n^2$
$\frac{2^n}{n^2} + \frac{2}{n} \geq c_1$
$for\ n = 1$
$2 + 2 \leq c_1$
$4 \geq c_1$
$for\ n = \infty$
$\lim_{n \to \infty} \frac{2^n}{n^2} = \infty$
$\infty + 0 \leq c_1$
$\infty \leq c_1$ there does not exists a real positive number greater than infinity.

$\mathbf{2n + 4^{\log_2 n} - 5 = \Theta(n^2)}$
True
$c_2 n^2 \leq 2n + 4^{\log_2 n} - 5 \leq c_2 n^2$
$c_2 n^2 \leq 2n + 2^{\log_2 n^2} - 5 \leq c_2 n^2$
$c_2 n^2 \leq 2n + n^2 - 5 \leq c_2 n^2$
$c_2 \leq 2/n + 1 - \frac{5}{n^2} \leq c_2$
$for\ n_0 = 4$
$c_2 \leq \frac{2}{4} + 1 - \frac{5}{4} \leq c_2$
$c_2 \leq \frac{1}{4} \leq c_2$
$for\ n = \infty$
$c_2 \leq 0 + 1 - 0 \leq c_2$
$c_2 \leq 1 \leq c_2$

Question # 2 **[1 marks]**

Show the correctness of following bubble sort algorithm using Loop Invariant. First explain the main condition of Loop Invariant followed by main steps to prove Loop Invariant.

```
BUBBLESORT(A)
    for i = 1 to A.length - 1
        for j = A.length downto i + 1
            if A[j] < A[j - 1]
                exchange A[j] with A[j - 1]
```

Solution:

Loop Invariant Condition: At the end of i iteration right most i elements are sorted and in place.

Loop invariant: At the start of each iteration of the **for** loop of lines 1-4, the subarray $A[1..i-1]$ consists of the $i-1$ smallest elements in $A[1..n]$ in sorted order. $A[i..n]$ consists of the $n-i+1$ remaining elements in $A[1..n]$.

Initialization: Initially the subarray $A[1..i-1]$ is empty and trivially this is the smallest element of the subarray.

Maintenance: From part (b), after the execution of the inner loop, $A[i]$ will be the smallest element of the subarray $A[i..n]$. And in the beginning of the outer loop, $A[1..i-1]$ consists of elements that are smaller than the elements of $A[i..n]$, in sorted order. So, after the execution of the outer loop, subarray $A[1..i]$ will consists of elements that are smaller than the elements of $A[i+1..n]$, in sorted order.

Termination: The loop terminates when $i = A.length$. At that point the array $A[1..n]$ will consists of all elements in sorted order.

Question # 3 **[1.5 marks]**

(a) What is meant by Design and Analysis of Algorithms?
(b) List two topics in Computer Science that are more important than studying computer program performance.
(c) Write down the formal definition of Small-Oh Notation i.e. in terms of f(n) and g(n)

Solution

(a) The analysis of algorithm is the theoretical study of computer program performance and resource usage. Algorithm design include creating an efficient algorithm to solve a problem in an efficient way using minimum time and space.
(b) Correctness, Security, Stability etc
(c) $o\big(g(n)\big) = \{f(n) \colon \exists c \text{ } and \text{ } n_0 \wedge c, n_0 > 0 \wedge f(n) < g(n) \forall n \geq n_0$

Question # 4 **[4 marks]**

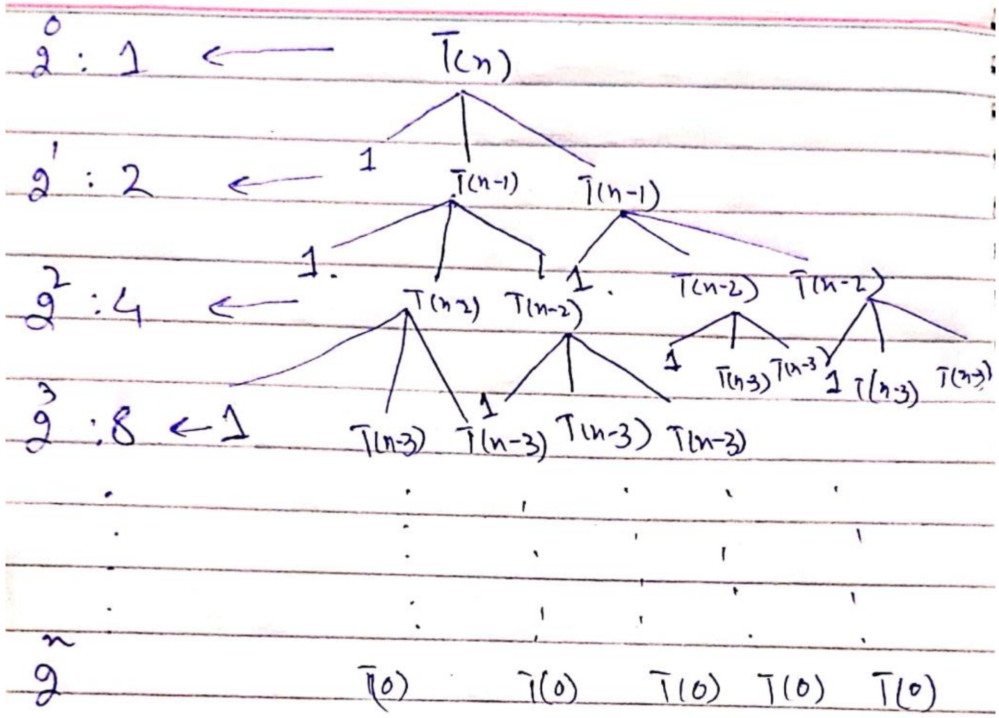
```
#include <iostream>
#include <unordered_map>
using namespace std;
// Function to find frequency of each element in a sorted array
void findFrequency(int arr[], int n, unordered_map<int, int>
&count)
{
	// if every element in the subarray arr[0..n-1] is equal,
	// then increment the element count by n
	if (arr[0] == arr[n - 1]) {
		count[arr[0]] += n;
		return; }
	// divide array into left and right sub-array and recur
	findFrequency(arr, n/2, count);
	findFrequency(arr + n/2, n - n/2, count);
}
```

```
// Find Frequency of each element in a sorted array containing duplicates
int main()
{	int arr[] = { 2, 2, 2, 4, 4, 4, 5, 5, 6, 8, 8, 9 };
	int n = sizeof(arr) / sizeof(int);
	// find frequency of each element of the array and store it in map
	unordered_map<int, int> map;
	findFrequency(arr, n, map);

	// print the frequency
	for (auto &p: map) {
		cout << p.first << " occurs " << p.second << " times\n";
	}
	return 0;
}
```

Question # 5 **[4.5 marks]**

$T(n) = 2T(n-1) + 1$



$= 2^0 + 2^1 + 2^2 + 2^3 + 2^4 + \cdots 2^n$

$a + ar + ar^2 + \cdots ar^n = \dfrac{a(r^{n+1}-1)}{r-1}$ ← Gp Series

$= 2^{n+1} - 1 = O(2^n)$

$T(n) = 2T(n-1) + 1$

$= 2[2T(n-2) + 1] + 1$

$4T(n-2) + 2 + 1$

$= 4[2T(n-3) + 1] + 2 + 1$

$= 8T(n-3) + 4 + 2 + 1$

$= 2^3 T(n-3) + 2^2 + 2^1 + 2^0$

Repeat K Times.

$= 2^K T(n-k) + 2^{(k-1)} + 2^{(k-2)} + 2^{(k-3)} + \cdots 2^1 + 2^0$

$n - k = 0 \quad ; \quad n = k.$

$= 2^k T(0) + 2^{k-1} + 2^{k-2} + \cdots 2^1 + 2^0$

$= 2^k + 2^{k-1} + 2^{k-2} + \cdots 2^1 + 2^0$

$= 2^n + 2^k - 1$

$= \Theta(2^n)$

(b) T(n) = 2T(n/4) + n

Let Guess #1: T(n) = O(n)

Need T(n) <=cn for some constant c > 0

Assume T(n/4) <= cn/4
Thus, T(n) <= 2cn/4 + n = n(c+2)/ 2

Since n(c+2)/2 <= cn for all c=1, n_0 = 1

Guess is correct

Now Check Guess #2 T(n) = O(logn)

Need T(n) <=clogn for some constant c>0

Assume T(n/4) <= clogn/4
Thus, T(n) <= 2clogn/4 + n = ½ clogn + n

Since ½ clogn + n <= cn Statement is False, thus correct complexity is O(n)

(c) T(n) = 7T(n/3) + n^2

Here a = 7,
b = 3
d = 2

Since a < b^d, Case 1 will be applied, Thus, T(n) = O(n^d) = O(n^2)

BEST OF LUCK