Design & Analysis of Algorithms
Mid-Term Examination (Fall 2022)

Reg. No: ________________ **Time: 90mins**

Q1 (10 + 10)

a) Solve the following recurrence, with all the steps and details and find the complexity of the recurrence.

$$T(n) = \begin{cases} 8T\left(\frac{n}{2}\right) + n^2 & ; if\ n > 1 \\ T(1) = 1 & ; if\ n = 1 \end{cases}$$

Ans:

T(n) = 8T(n/2) + n2 (A)

= $8[8T(n/2^2) + (n/2)^2] + n^2$

T(n) = $8^2T(n/2^2) + 2n^2 + n^2$ (B)

= $8^2[8T(n/2^3) + (n/2^2)^2] + 2n^2 + n^2$

T(n) = $8^3T(n/2^3) + 4n^2 + 2n^2 + n^2$ (C)

= $8^3[8T(n/2^4) + (n/2^3)^2] + 4n^2 + 2n^2 + n^2$

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

.

.

.

T(n) = $8^kT(n/2^k) + 2^{(k-1)}n^2 + 2^{(k-2)}n^2 + \ldots + 2^3n^2 + 2^2n^2 + 2^1n^2 + 2^0n^2$ (G)

Let $n/2^k = 1; n = 2^k$

T(n) = $(2^3)^kT(1) + n^2[2^{(k-1)} + 2^{(k-2)} + \ldots + 2^2n^2 + 2^1n^2 + 2^0n^2]$

T(n) = $(2^3)^kT(1) + n^2[2^{(k-1)} + 2^{(k-2)} + \ldots + 2^2n^2 + 2^1n^2 + 2^0n^2]$

GEOMETRIC SERIES

$[2^{(k-1)} + 2^{(k-2)} + \ldots + 2^2 + 2^1 + 2^0] = (2^k - 1)/(2 - 1) = n - 1$

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

= $O(n^3)$

Marks break-up

A+B+C+D --- 4 marks

G --- 3 marks

After G --- 3 marks

Solution Mid Term AA F22

Reg. No: ____________________ **Time: 90mins**

b) Write exact frequency count of each line and give asymptotic complexity of the following code snippet. Show Details.

```
for (int i = n; i > 0; i=i-2) {
        for (int j=1; j < i; ++j)
                a = a + b * 2;
}
```

(**Note:** It is not required to give exact operations count of each line.)

Ans: The worst-case occurs when n is odd.

for (int i = n; i > 0; i=i-2) {	c1	$(n+1)/2+1$	//L1
for (int j=1; j < i; ++j)	c2	$1+\frac{(n+1)(n-1)}{4}+\frac{(n+1)}{2}$	//L2
a = a + b * 2;	c3	$1+\frac{(n+1)(n-1)}{4}$	//L3
}			

T(n) = O(n^2)

L1: up to 2 marks
L2: up to 4 marks
L3: up to 2 marks

Details of L2 & L3: 2 marks

Q2 (5 + 15)

Given a sorted array, **Arr**, of size N, which contains only two distinct elements **x & y**, where **x < y**, it is required to find the count of **y** in the array **Arr**. **The Time Complexity of the algorithm shall be O($\log_2$n).**

a) Briefly describe your algorithm in words.
b) Write a C++ function that returns the count of **y** elements.

Example:

Arr:	2 2 2 2 2 2 2 5 5 5 5
Returned Value:	4
Arr:	-12 -12 -12 7 7 7 7 7 7 7 7 7
Returned Value:	9

Ans: (a)

If the explanation is understandable, in readable English, according to code given in part (b), even the code is not correct --- 3 marks

Ans: (b)

```
int countGreater (const int* arr, int size) {
   int start {}, end {size};
   int key = arr[size-1];
   while (start < end) {
      int mid = start + (end - start)/2;
      if (arr[mid] < key)
         start = mid + 1;
      else
         end = mid;
   }
   return size - start;
}
```

Correct header --- 2 marks
Correct O(n) code --- additional ~~3~~ 5 marks
Correct O(lgn) code --- additional 13 marks

Q3 (5+5+5+5)

```
quickSort(Arr, l, r)
  1. if l < r {
  2.     let p = PARTITION(Arr, l, r)
  3.     quickSort(Arr, l, p-1)
  4.     quickSort(Arr, p+1, r)
  5. }
```

```
PARTITION(Arr,l,r)
    let pivot = Arr[r]
    let i = l - 1
    for j = l to r - 1 {
        if Arr[j] <= pivot {
            i = i + 1
            exchange Arr[i], Arr[j])
        }
    }
    exchange(Arr[i+1], Arr[r])
    return i + 1
```

Let Arr = 8 29 7 11 18 6 2 9 1 15 be used for answering the following parts.

a) Give the count of least(minimum) number of elements that are in their correct sorted position when Line 3 of ***quickSort(...)*** is reached (but not executed) for the fourth time.
   Ans: 4

b) What are the contents of ***Arr*** when Line 3 of ***quickSort(...)*** is reached (but not executed) for the third time.
   Ans: 1 7 6 2 8 9 11 15 18 29

c) What are the values of ***p*** and ***r*** when Line 4 of ***quickSort(...)*** is reached (but not executed) for the first time.
   Ans: p=0 or 1 ; r=6 or 7

d) Write the recurrence equation for the worst case time complexity of ***quickSort(...)***. (Solution of recurrence equation is not required.)

   Ans: $$T(n) = \begin{cases} c; & if\ n = 1\ or\ n = 0 \\ T(n-1) + T(0) + O(n); & if\ n > 1 \end{cases}$$