Name: ____________________ Roll #: ____________________ Section: _____________

National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
Program:	**BS(Computer Science)**	**Semester:**	**Fall 2020**
Duration:	**90 Minutes**	**Total Marks:**	**40**
Paper Date:	**21-Oct-20**	**Weight**	**12.5%**
Section:	**ALL**	**Page(s):**	**5**
Exam:	**Midterm 1**		

Instruction/Notes: Attempt the examination on the question paper and write concise answers. You can use extra sheet for rough work. Do not attach extra sheets used for rough with the question paper. Don't fill the table titled Questions/Marks.

Question	1	2	3	4	5	Total
Marks	/ 5	/8	/5	/10	/12	/40

Q1) [5 marks] Selection Sort is an O(n^2) algorithm that works by repeatedly swapping the next element in the array with the next minimum element. A pseudo-code is given below:

```
SelectionSort(A, n)
     FOR i ← 1 to n-1
          m ← i //assume i is the minimum index
          FOR j ← i+1 to n
                    IF(A[j] < A[m])
                         m←j //update minimum index
     swap(A[i], A[m])  //swap min element with the ith element.
```

Is this algorithm, as described above, a stable sorting algorithm? Answer Yes or No. Then justify your answer in two lines.

Not stable:
If we run it on following array, 9a and 9b will not be in orignal order

Input: {2, 9a, 17, 18, 9b, 13, 4}
After first iteration of outer for loop: {2, 9a, 17, 18, 9b, 13, 4}
After second iteration of outer for loop: {2, 4, 17, 18, 9b, 13, 9a}
After third iteration of outer for loop: {2, 4, 9b, 18, 17, 13, 9a }
After fourth iteration of outer for loop: {2, 4, 9b, 9a, 17, 13, 18 }
After fifth iteration of outer for loop: {2, 4, 9b, 9a, 13, 17, 18 }
After sixth(last) iteration of outer for loop: {2, 4, 9b, 9a, 13, 17, 18 }

Q2) [8 marks] You are implementing a class Set, where each set contains an array of unique ASCII characters. You wish to add the union and intersection methods to your Set class. What is the fastest asymptotic running time in which these functions can be performed between two such sets of size n each? First give the answer in terms of big-Oh, then explain your answer in a few lines.

use count sort on both sets for sorting in linear time and then use merge routine for taking union and intersection.

OR

Use counting table of count sort for intersection (indices where we get a count of 2) and union (taking all indices where we get a positive count).

Q3) [5 marks] The following line is a key part of the Merge Sort algorithm:

mid ← (left + right) /2

Suppose Merge Sort was applied to an array of size n. Then the above line will be executed approximately how many times? Encircle the correct answer below, then justify your answer in a few lines.

i) O(nlgn) times
ii) O(n) times
iii) O(lgn) times
iv) O(1) times.

O(n) times

This line is executed once in each reccrsive call. Each reccrsive call is represented by a node in reccursion tree.

level	Number of nodes
0	1
1	2
2	$4 = 2^2$
3	$4 = 2^3$
. . .	
$K = \log_2 n$	$(2)^k$

$$\sum_{i=0}^{\log_2 n} 2^i = (2^{\log_2 n+1} - 1) / (2\text{-}1) = (2.n - 1) = \Theta(n)$$

Q4) Consider the following recursive algorithm:

```
        StrangeSummation(A, p , r, sum) //sum is passed by reference
                IF(p<r){
                     n ← r - p + 1
                    StrangeSummation (A, p, p + n/3, sum);
                    StrangeSummation (A, p + 2n/3, r, sum);

                    FOR i← p to r
                           sum←sum+A[i];
               }
```

You may assume that $n=3^k$, where k= 0, 1, 2, …

i) [4 marks] Write the recurrence for the time function T(n).
ii) [6 marks] Solve your recurrence and derive a Big-Oh bound.

recurrence: T(n) = 2T(n/3) + O(n)

The reccurence tree will have $\log_3$ n levels.

level	Number of computations
0	n
1	n/3 + n/3 = 2n/3
2	4n/9
3	8n/27
. . .	
K = $\log_3$ n	$n(2/3)^k$

Total computations at all levels of tree = n (1 + 2/3 + $(2/3)^2$ + $(2/3)^3$ ….. $(2/3)^{\log_3 n}$)

$$= n\sum_{i=0}^{\log_3 n} 2/3^i \le n\sum_{i=0}^{\infty} 2/3^i = n\ (1/\ (1-2/3)) = O\ (n)$$