COMSATS University Islamabad Main Campus.
Department of Electrical and Computer Engineering

CSC-211 Data Structures and Algorithms

Terminal Exam FA 2023

Class: **BCE-3B** Date: **09 Jan. 2024**

Time Allowed: **180 Mins** Marks: **50**

Question 1. (C2-PLO1)

Part 1: Answer the following brief questions. **(5 marks)**

a. What is a Complete Binary Tree? How are heaps use Complete Binary Trees in their array implementation?

b. How are stacks used to solve the Towers of Hanoi problem?

c. What are the two types of self-balancing binary search trees that we studied in class? How do they keep balance in face of addition and deletion of nodes?

d. What is the minimum number of levels that a binary search tree with 200 nodes can have?

e. What do the Big-O, Omega (Ω) and Theta (θ) notations for algorithm analysis (time complexity) signify?

Part 2: **(5+5+5 marks)**

i. Create a ***Max Heap*** using the following sequence of data.

ii. Also draw the array representation of this min-heap.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Num	39	48	38	26	115	69	18	120	52	11	128	57	90	1	5	24

Use your (**Roll No%10**) as index to find the starting point. E.g., if your Roll number is 178, then your starting index will be 178%10 = 8. The first number you will enter in the tree will be 52.

iii. Draw a weighted graph showing the road network between eight major cities of Pakistan. Choose an appropriate parameter for the weight of the links. Assume values for that parameter for each link. Explain how this graph can be used to plan a route from one city to another.

Q6. Write a C function to generate all permutations of a given string. You are required to implement the function with the following prototype: **(C3-PLO1) (15 marks)**

```
void stringPermutations(char *str, int start, int end);
```

This function should use recursion to generate and print all permutations of the string ***str***. The parameters ***start*** and ***end*** denote the starting and ending indexes of the substring of ***str*** that you are currently permuting.

Additional Requirements:

- You may write any additional helper functions if needed, but ensure that the main functionality is implemented in the ***stringPermutations()*** function as per the given prototype.
- Your function should handle strings of varying lengths gracefully.
- Comment on your code to explain your logic, especially the recursive calls and the base case.

Note: For this exam, you can assume that the function:

```
void swap(char * x, char * y);
```

is already implemented, which swaps the characters pointed to by x and y.

Algorithm:

Base Case: If the string length is 1, there's only one permutation – the string itself.

Recursive Case:

For each character in the string:

- Fix that character at the first position.
- For the remaining characters, recursively find permutations.
- After the recursive call, swap back the characters to maintain the original string order. This step is crucial when implementing the algorithm in place to avoid altering the original string's order.

**** End of Document ****