United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Final Exam: Summer 2024
Course Code: CSE 2215, Course Title: Data Structures and Algorithms 1
Total Marks: 50 Duration: 2 hours

Answer all questions. Marks are indicated in the right side of each question.

[Any examinee found adopting unfair means will be expelled from the trimester/program as per UIU disciplinary rules.]

---

1. a. Convert the following infix expression into postfix using a STACK.
   Infix expression: (A×B+C)÷(D−E)×(F+G)−H **[3]**
   b. Evaluate the postfix expression, AB∗CD+÷EF+∗
   for A=6, B=4, C=8 , D=4, E= 5,and F=3 using a STACK. **[3]**
   c. Find a recursive algorithm for TOWER OF HANOI using one intermediate pillar/peg and show simulation for n = 3, where n is the number of disks. **[4]**

2. You are to store the exam marks of 12 students in DSA I using a Binary Search Tree (BST). However, the marks that you are supposed to store in that BST are given in two different sequences, as given below:

   Sequence 1: 50, 30, 70, 15, 40, 75, 74, 95, 10, 40, 60, 62
   Sequence 2: 10, 15, 30, 40, 40, 50, 60, 62, 70, 74, 75, 95

   Based on this context, answer questions **2a-2d**.

   a. Draw a BST from given sequence 1 (Tree 1) and sequence 2 (Tree 2). **[2+2]**
   b. Now, delete the following nodes one after the other from Tree 1. Redraw the tree after each deletion. **[2+2]**
      I. 50 II. 70

   c. Calculate the number of moves you'll need to find the minimum and maximum value in Tree 1 (before deletion) and Tree 2.

      Discuss in which cases these operations (finding min and max) are more efficient by comparing the average and worst-case time complexities. Use these two trees as examples. **[3]**
   d. In a world where we're always obsessed with who became first (or has the highest marks), you decided to give some limelight to the second person as well!

      Design an algorithm to find the second-highest value from a BST. Note that you may not delete any nodes from the BST in the process.

      Writing the pseudocode or C/C++ code for this particular function would be enough. **[3]**

3. You are an explorer in a labyrinth with 8 rooms arranged in a grid-like manner. Each room is connected to its neighboring rooms by corridors. Generally, you can only move from **left to right and top to bottom** (See the figure below). For example, you can move from Room 1 to Room 2 but not Room 2 to Room 1. Similarly you can move from Room 4 to Room 7 but not from Room 7 to Room 4.
   Again, there are some rooms that have **locked doors** that you need to unlock by visiting certain other rooms first. Rooms **3, 6, and 8** are locked.
   To unlock **Room 3**, you must visit **Room 5** first.
   To unlock **Room 6,** you must visit **Room 3** first.
   To unlock **Room 8**, you must visit **Room 6** first.
   Note that, you can directly go from the prerequisite rooms to the locked rooms; you don't have to follow the left to right or top to bottom condition for these rooms.
   Now answer the following questions based on this scenario.

Room 1	Room 2	Room 3
Room 4	Room 5	Room 6
Room 7	Room 8	

   a. Represent the rooms and their connections in a **directed graph**. Each vertex should represent a room, and each directed edge represents the connections or accessibility between rooms, including the condition that one room must be visited before another (i.e., the unlocking conditions). Draw this graph. **[4]**
   b. Perform a **topological sort** of the rooms considering the unlocking conditions. What order should you visit the rooms in to unlock all the required doors? **[3]**
   c. Simulate the **depth-first search** starting from Room 1 to Room 8. Describe the path taken by DFS and identify the order in which rooms are visited. In how many moves did you reach Room 8? **[3]**
   d. Simulate a **breadth-first search** starting from Room 1 to Room 8. Describe the path taken by BFS and compare it with the DFS traversal by calculating the number of moves. Which search is more suitable for this scenario? **[4]**

4.

   a. Suppose you have a max-heap in an array, and you have to sort the array in the descending order using heapsort algorithm. Will you have to call the Build-Heap function before you start sorting? Explain why, or why not. **[3]**

   b. Apply heapsort algorithm on the following array to sort it in the ascending order. You have to show the heap after the Build-Heap operation, and after each further Heapify call. **[6]**

      23, 56, 11, 87, 34, 75

   c. Suppose you have a priority queue built over a max-heap. We know that if the priority of any element in the queue increases, we have to call the Increase-Key procedure. But it is possible that the priority of some element may decrease. How can you handle this case? Can you use any known function related to heap or priority queue? **[3]**