Data Structures (IMT+IMG 2023)
Mid Semester

Note: Lengthy paper; please be brief in your answers

1. A little cumbersome sorting. Read all parts before answering the first part. In the following questions, if you are using a struct or a class, please give the code for the same as well.

i. Implement the enqueue, dequeue and front operations of a queue using a circular array implementation using a C/C++ program. [2+2+1]

ii. You are given an array of *n* integer elements. Find the minimum and maximum element of the array using **a recursive** approach only. [2]

iii. You are given an array of *n* integer elements and an integer *k* (1<*k*<*n*). Make an array of *k* queues. Divide the range (minimum value to maximum value) of values in the array into *k* parts and assign 1 queue for each part. Return the array of queues. *E.g.* [2]

- Suppose the array is (5, 8, 1, 2, 7, 9, 0, 3) and *k*=3.
- The range of values [0, 9] is divided into 3 parts [0,3), [3,6) and [6,9]. The first queue has values from 0 (inclusive) to 3 (exclusive). The second queue has values from 3 (inclusive) to 6 (exclusive). The third queue has values from 6 (inclusive) to 9 (inclusive).
- The first queue will have all elements from [0,3) which are (1, 2, 0).
- Second queue will have all elements from [3,6) which are (5, 3)
- Third queue will have all elements from [6,9) which are (8, 7, 9)

iv. Write the C/C++ code for the quick sort algorithm. [3]

v. Discuss the complexity of quick sort algorithm for the best and worst cases. [1]

vi. Write the C/C++ code for a sorting algorithm that divides the sub-array into *k* parts, equally divided by the range of values of the sub-array, using the solution to (iii) above. Memory complexity is not a constraint to be considered when formulating the solution. [3]

2. Let us set a random question paper. Read all parts before answering the first part. In the following questions, if you are using a struct or a class, please give the code for the same as well. If you cannot attempt the question for the asked data type, attempt using integers instead for partial marks.

i. Show the node definition for a linked list that stores attributes "*question text* (string)", "*chapter number* (integer)" and "*marks* (integer)". This may not be the same as the node for an integer linked list. *Chapter number* is an integer between 0 and *c*, where the textbook has *c*+1 chapters numbered sequentially using integers. [1]

ii. Write a C/C++ code for a function that inserts a node after a node pointer *p* in the linked list. [1]

iii. Write a C/C++ code for a function that inserts a node at the beginning of a linked list. [1]

iv. Suppose you have a linked list called "*question paper*" of nodes of the type described in (i). Calculate the total marks for all nodes in the linked list. [1]

- **Or** for partial marks, given a linked list, calculate the sum of all elements of the linked list.

v. Suppose you have a linked list of nodes of the type described in (i) called "*question paper*" arranged in a sorted order as per the chapter number. You also have all the data of a new question to be inserted into the linked list. Place the new question maintaining the sorted order. In case there are multiple questions of the same chapter, insert the new question towards the end. [2]

- **Or** for partial marks, given a sorted linked list of integers, add a new element in the linked list maintaining the sorted order.

vi. Suppose an array of linked lists exists of the node type (i) called "*question bank*". Each chapter number is a cell of the array. All questions from the same chapter are arranged as a linked list of questions. Given all data of a new question, insert the question into the question bank. You have the option to add the question at the start, middle, or rear of the existing questions from the chapter. [1]

- **Or** given an array of linked lists, insert a node at the i^{th} linked list in the array. You may insert the node at the start, middle, or rear of the linked list, as per your choice.

vii. Suppose the function *randInt(k)* returns a random integer between 0 and *k*. Select a random question from a random chapter from the question bank linked list. Add the same question to the question paper if and only if the total marks do not exceed 30. Repeat till a random question paper worth 30 marks is set. To avoid insertion of a question twice, delete any question that is inserted into the question paper. You may assume a standard function *size* that returns the size of a linked list. [3]

3. Applied Stacks

i. Implement a stack using queues [2]

ii. Show the evaluation of the following postfix expression: 1 5 + 6 * [1]

iii. Convert the following infix to postfix using stacks: (4+6) * 2 +3 [1]