Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Total
10	6	8	6	10	20	20	20	100

CMPE 242 – Final Exam

Date: 10.06.2021 13:30-15:30

Name:
Student ID:

You have 120 minutes for this exam. The exam is closed book, closed notes, except that you are allowed to use an **A4-size, double sided, handwritten** cheat sheet. You have additional 15 minutes to scan your answers as a pdf file, and upload your answers to Moodle.

Algorithm Analysis

1. (10 points) What is the order of each of the following tasks in the **worst** case of the **best** algorithm for the task (in Big-O notation)?

    - Searching a sorted array of *n* integers for a particular value. *Answer:* _______
    - Searching a linked list of *n* integers for a particular value. *Answer:* _______
    - Searching an unsorted array of *n* integers for a particular value. *Answer:* _______
    - Inserting a new value into a sorted array of *n* integers. *Answer:* _______
    - Sorting an array of *n* integers using key comparisons. *Answer:* _______
    - Inserting *n* integers into an empty AVL binary tree. *Answer:* _______
    - Deleting all *n* items, one by one, from a 2-3 tree. *Answer:* _______
    - Given an AVL tree of *n* nodes, visit all nodes of the tree according to the post-order traversal. *Answer:* _______
    - Deleting the max value from a max heap of *n* integers. *Answer:* _______
    - Checking whether or not an array of *n* integers is a heap. *Answer:* _______

Context

2. (6 points) Match each of the following applications with the **best** data structure or algorithm that play an important role for implementing them. Use each data structure / algorithm only once.

Applications	Data Structures and Algorithms
___ Event-driven simulations	(a) Sorting
___ Symbol tables with ordered operations	(c) Priority Queues
___ Windows folders	(d) Binary Search Trees
___ Music player play lists	(e) Stacks
___ Viewing emails in day order	(f) Queues
___ Implementing Back button on browser	(g) General Trees

Priority Queues / Heaps

3. (8 points)

   a) Show the result of inserting 10, 12, 1, 14, 6, 5, 8, 15, 3 one at a time, to an initially empty **min-heap**, then removing two items. Note: show the intermediate steps and the resulting heap as a tree.

   b) Given the following arrays, which ones could represent a max-heap? Choose all that apply. Assume the indices start from 1.
   ___ {23, 17, 14, 6, 13, 10, 1, 12, 7, 5}
   ___ {23, 17, 14, 6, 13, 10, 1, 5, 7, 12}
   ___ {23, 17, 14, 7, 13, 10, 1, 5, 6, 12}
   ___ {23, 17, 14, 7, 13, 10, 1, 12, 5, 7}

Sorting

4. (6 points)

   a) We are sorting the below numbers by using quicksort algorithm. If we complete the first partition operation by choosing a pivot randomly, which statements could be true?

   Set: {7, 10, 6, 12, 14, 17, 16, 15}

   ____ The pivot could be either the 12 or the 14.
   ____ The pivot could be the 12, but it is not the 14.
   ____ The pivot is not the 12, but it could be the 14.
   ____ Neither the 7 nor the 9 is the pivot.

   b) Consider you are given an array of elements arr[5]= {7,6,5,4,3}, and you are asked to sort the array in ascending order. What are the steps of insertions done while doing insertion sort in the array? Show each major intermediate step in your answer.

Trees

5. (10 points) Create an AVL tree by using the below set of number step by step:

   {14, 20, 25, 13, 12, 18, 15}

   Then, delete 12 and check whether the updated tree is an AVL tree. If it is not an AVL tree, you need to make it an AVL tree by completing necessary operations step by step. Your solution should be clear and detailed.

Algorithm Design (Heaps)

6. (20 points) Please write a method that converts a maximum binary heap to a minimum binary heap. Your algorithm should run in linear time (i.e. **O(N)**).

   a) First explain how your method works in 2-3 sentences,

   b) Secondly, write the Java code for the method:

```
public static void convertHeap ( int[] a, int N)
{
```

   c) Explain the big-O complexity of your algorithm.

Algorithm Design (Trees)

7. (20 points) Given two values *k1* and *k2* (where *k1* < *k2*) and the *root* of a Binary Search Tree, write a **Java method** that trims a tree. The trimming operation is defined as **removing the leaves with keys smaller than *k1* and the leaves with keys larger than *k2*;** and keeping the rest of the leaves as well as all the internal nodes in the tree.

   Use the following Node declaration. You may assume that the keys are unique in the given tree.

   Note: you should not visit a tree branch unnecessarily if its keys are outside this range. Your answer will be graded on correctness, efficiency, clarity, and conciseness.

```
private class Node {
     private int key;       % key
     private Node left;     % left child
     private Node right;    % right child
}
```

   a) First explain how your method works in 2-3 sentences,

   b) Secondly, write the Java code for the method. The method should have a parameter of type `Node`, which is the reference to the root of the tree, and the two key values k1 and k2. It should return the new root as a result. (Hint: you may use additional helper methods for recursion.)

```
public static Node trim ( Node n, int k1, int k2)
{
```

   c) Explain the big-O complexity of your algorithm.

Algorithm Design (Context)

8. (20 points) Suppose that a linked list is formed from objects that belong to the class:

```
class Node {
   int item;      // An item in the list.
   Node next;     // Pointer to next item in the list.
}
```

Given a single linked list as input, write a **Java method** to sort it. Your algorithm must run **in time better than O(N^2).** You may only use constant size **(i.e. O(1)) additional memory** and your algorithm should work on the linked list itself (i.e. meaning you cannot allocate another array to temporarily hold the items. No points will be given if you use an extra array.).

a) First explain how your method works in 2-3 sentences. Explain which sorting algorithm you will modify.

b) Secondly, write the Java code for the method. The method should have a parameter of type `Node`, which is the reference to the head of the list (Note: you may use a helper function if necessary):

```
public static void sort(Node head ) {
```

c) Explain the big-O complexity of your algorithm.

<<< END OF EXAM. THANK YOU. >>>