CMPT225, Spring 2021

Midterm Exam - Sample

Name_____

SFU ID: |__|__|__|__|__|__|__|__|__|

Problem 1	
Problem 2	
Problem 3	
Problem 4	
TOTAL	

Instructions:

1. You should write your solutions directly in this word file, and submit it to Coursys. Submitting a pdf is also ok.
2. No late submissions, no exceptions
3. Write your name and SFU ID on the top of this page.
4. This is an open book exam. You may use textbooks, calculators, wiki, stack overflow, geeksforgeeks, etc. If you do, specify the references in your solutions.
5. Discussions with other students are not allowed. Posting questions online asking for solutions is not allowed.
6. The exam consists of four (4) problems. Each problem is worth 25 points.
7. Write your answers in the provided space.
8. Explain all your answers. Really, explain all your answers.

Good luck!

Problem 1 [25 points]

A. (3 points each) For each sentence decide whether it is True or False. **Write a brief explanation**.

1) Every recursive method can be written without making recursive calls.

2) Is the following definition legal of foo()? i.e., does it compile?

```
public int foo() { return null; }
```

3) Is the following definition legal of foo()? i.e., does it compile?

```
public Integer foo() { return null; }
```

4) Suppose we have two functions

```
public void foo(String s) { ...}
public void bar(String s) { ...}
```

Suppose the running time of foo $O(n^2)$ and the running time of bar is $O(n^3)$, where n is the length of the string. Is it true that foo() is faster than bar() on the input "0101".

B. (6 points) Consider the following function. Use big-O notation to express the running time of the following function on an array of length n with start=0 and end=n-1. **Explain your answer.**

```
public int f1(int array[], int start, int end) {
        if (start == end) {
                array[start]++;
        }
        else {
                int quarter = (end - start)/4;
                f1(array, start, end - 2*quarter);
                f1(array, start + 2*quarter, end);
                f1(array, start + quarter, end - quarter);
        }
}
```

C. (7 points) Write a non-recursive method equivalent to f2(`int n`) below whose running time is $O(n^2)$. **Explain your answer, and explain the running time.**

```
public int f2(int n) {
        if (n <= 2)
                return n;
        int sum = 0;
        for (int i = 1; i < n; i++)
                sum += f2(i) + f2(i-1);

        return sum;
}
```

Problem 2 [25 points]

In this problem use the following definition of Binary Tree. You may assume the classes have the standard getters/setters.

```
public class BTNode<T> {
    private T data;
    private BTNode<T> leftChild;
    private BTNode<T> rightChild;
    private BTNode<T> parent;

}

public class BinaryTree<T> {
    private BTNode<T> root;

}
```

A. (6 points) Write a method for the class Binary Tree that returns the number of leaves in the tree. The running time should be O(size of the tree).
Explain your algorithm and running time.

```
public int numberOfLeaves() {









}
```

B. (7 points) Write a method that returns an ArrayList with all the values in the tree. The running time should be O(size of the tree). **Explain your algorithm and running time**.

```
public ArrayList<T> preOrderArrayList() {









}
```

C. (12 points) Write a method for the class Binary Tree that gets another tree and checks if the trees have the same preOrder traversal. The running time should be O(n) in the worst case, where n is the size of the smaller tree. For example, if one tree has n vertices, and the other has n^2 vertices, the running time should be O(n). **Explain your algorithm and running time**.

```
public boolean samePreOrder(BinaryTree<T> tree) {

}
```

Problem 3 [25 points]

A. (7 points) Let A = [1, 4, 2, 9, 3, 8, 5, 0]. Apply the build-minHeap algorithm on A using the linear time algorithm we saw in class, and draw the tree representation of the heap. **Draw the intermediate steps.**

Recall the buildHeap algorithm:

```
Treat the array as a complete binary tree
For each vertex v starting from the bottom
    Apply heapify(v)
```

B. (4 points) Apply removeMin() on the heap obtained in part A, and draw the resulting heap.

C. (4 points) Add 6 to the heap obtained in part B, and draw the resulting heap.

D. (4 points) If in the buildHeap() algorithm we iterate from top down, and heapify (push down) each node, will we get a heap in the end? If yes, prove it. If no provide a counter-example

E. (6 points) Write an algorithm that gets a Binary Search Tree of ints and creates a minHeap with the same values in O(n) time, where n is the size of the tree. The heap should be represented as an array.

Problem 4 [25 points]

A. (8 points) Write an algorithm that gets a root of a Binary Search Tree of ints and a number k, and returns a List with all numbers in the tree with value at most k. Write the algorithm as efficiently as possible. Ideally the running time should be O(length of the output + height of tree)
Explain your answer and explain the running time.

```
public ArrayList<Integer> valuesAtMostK(BTNode<Integer> bstRoot, int k) {









}
```

B. (8 points) Write an algorithm that gets two Stacks and checks if they are equal. In the end of the algorithm the stacks should be in their original state.

```
public <E> boolean equalStacks(Stack<E> s1, Stack<E> s2) {









}
```

C. (3 points each)

- Convert the following expression from infix to prefix: ((7 - 1) + (8 * (6 / 2)))
- Convert the following expression from prefix to infix: / + 4 - 10 2 + 3 0
- Convert the following expression from prefix to postfix: / 14 - - 10 2 / 3 3