CMPT 225 D100, Spring 2023

Final Exam
April 18, 2023

Name________________________

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

Problem 1	
Problem 2	
Problem 3	
Problem 4	
TOTAL	

Instructions:

1. Duration of the exam is 180 minutes.
2. Write your full name and SFU ID **clearly**.
3. This is a closed book exam, no calculators, cell phones, or any other material.
4. The exam consists of four (4) problems, each worth 25 points
5. Write your answers in the provided space.
6. There is an extra page at the end of the exam. You may use it if needed.
7. You may write helper functions if needed
8. Explain all your answers.
9. Really, explain all your answers.

Good luck!

Problem 1 (Java syntax, Big-O notation) [25 points]

A. (5 pts) Consider the following function.

```
public static void do_something(int n) {
  ArrayList<String> ar = new ArrayList<String>();
  for (int i = 0; i < n; i++)
    ar.add(String.valueOf(i)); // adds the strings "0", "1", "2",...
  for (int i = 0; i < n; i++)
    ar.remove(0); // remove the element in index 0
}
```

What is the running time of this function? Use Θ notation to express the running time.

B. (5 pts) Prove that the running time of the function foo(n) is $O(n^{\log_3(5)\approx 1.465})$.

```
public static int foo(int n) {
    if (n>=4)
        return foo(n/2)+foo(n/3)+foo(n/4);
    return 1;
}
```

[Bonus: 5 pts] Prove that the running time is $O(n^c)$ for some $c<\log_3(5)$.

C. (5 pts) Consider the class IntString.

```
public class IntString {
  private int n;
  private String s;

  @Override
  public int hashCode() {
    if (s.length()>0)
      return (n*s.hashCode()) % 1000;
    else
      return n % 1000;
  }

}
```

Write the method findHashCollison() that gets an instance of the class, and returns another instance that has the same hashCode.
You may assume the class has the standard constructors/getters/setters.

```
public static IntString findHashCollision(IntString is) {





  }
```

D. (10 pts) List 5 public methods of the class java.util.LinkedList? Briefly explain their running time.

Problem 2 (Binary Trees) [25 points]

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

```
public class BTNode {
    private int data;
    private BTNode leftChild;
    private BTNode rightChild;
    private BTNode parent;
}

public class BinaryTree {
    private BTNode root;
}
```

A. (10 pts) Draw the AVL tree obtained by inserting the following sequence of numbers: 8,7,6,5,4,3,2,1,0. Draw some intermediate trees to show your work.

B. (15 pts) Write a method that gets a Binary Tree and checks if it is a Binary Search Tree, i.e., every node is >= vertices in the left subtree, and <= vertices in the right subtree. The running time must be O(size of tree).
Explain the idea of the algorithm and the running time.

```
public static boolean isBST(BinaryTree tree) {









}
```

Problem 3 (Quick sort and Selection algorithm) [25 points]

A. (12 pts) Write a function that gets an array of ints and a number k, and rearranges the array so that all numbers <=k appear in the array before all numbers >k.
The function needs to run in time O(ar.length), and use only O(1) additional memory.
Explain the idea of the algorithm.

```java
public static void rearrange(int[] ar, int k) {








}
```

B. (6 pts) Show the execution of the Selection Algorithm that searches for the median of the array A = [9,16,5,10,6,12,0,6,5,4,6,9,8,1,2]. State explicitly all recursive calls.

C. (7 pts) Consider the variant of Selection Algorithm, where we partition the input array into n/9 tuples each of size 9 (instead of n/5 tuples of size 5 we saw in class).
What will be the running time of the algorithm? Write the recursive formula for the running time, and then compute the closed formula.

Problem 4 (MinHeap) [25 points]

A. (7 pts) Apply buildMinHeap on the array A=[4,10,8,1,5,6,9,2,15,0]. Show some intermediate steps by drawing the corresponding tree and the array.
Draw the final result as a tree and as an array.

B. (3 pts) Apply removeMin() from the heap obtained in part A. Draw the result as a tree and as an array.

C. (15 pts) Write a function that gets a MinHeap of n ints given as an array, and a parameter k (0<k<=n), and returns the k smallest numbers in the heap.
The heap itself remains unchanged.
- For 5 points write a function whose running time is $O(2^k)$
- For 10 points write a function whose running time is $O(k^2)$
- For 15 points write a function whose running time is $O(k \log(k))$

Before writing code, explain the idea of the algorithm. You may assume you have the standard functions of the class MinHeap: getMin(), removeMin(), getLeftChild(), getRightChild(), getParent(), buildHeap(), heapify()...

```
public static Collection<Integer> getMinK(int[] heap, int k) {
```

Extra page

Master Theorem

Let $T(n) = a\,T(n/b) + f(n)$, $T(1) = O(1)$

Define $c = \log_b(a)$

- If $f(n) = O(n^d)$ for $d<c$, then $T(n) = \Theta(n^c)$
- If $f(n) = \Omega(n^d)$ for $d>c$, then $T(n) = \Theta(n^d)$
- If $f(n) = \Theta(n^c)$, then $T(n) = \Theta(n^c \log(n))$