15-121
Fall 2019 Final Exam
December 8, 2019

Name:

Andrew ID:

- You may not use any books, notes, or electronic devices during this exam.
- Show your work on the exam to receive credit.
- You may complete the problems in any order you'd like; you may wish to start with the free response problems, which are worth most of the credit.
- All code samples run without crashing unless we state otherwise.
- Assume any imports are already included as required.

Don't write anything in the table below.

Question	Points	Score
1	13	
2	9	
3	8	
4	10	
5	4	
6	6	
7	10	
8	10	
9	20	
10	10	
11	0	
Total:	100	

1. **Short Answer**

   Answer each of the following *in just a few words.*

   (a) (2 points) If you implement a stack with an `ArrayList`, can all the operations be $O(1)$? If so, describe how. If not, describe why not.

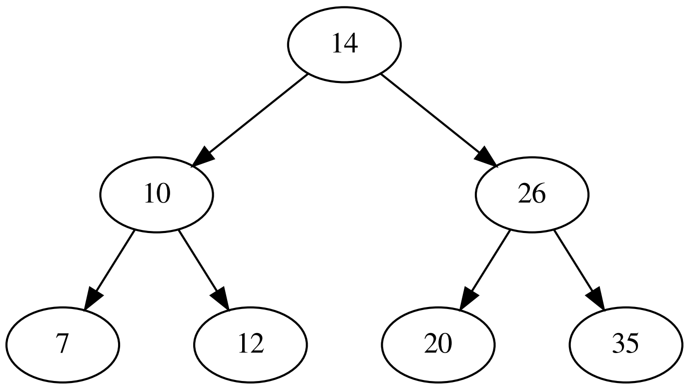
   (b) (2 points) If you implement a stack with a `LinkedList`, can all the operations be O(1)? If so, describe how. If not, describe why not.

   (c) (2 points) The standard convention in Java is to declare all of your instance variables private. Why is this a good idea?

   (d) (1 point) Consider a binary tree stored in an array as discussed in the notes and class. If a node is stored at index 7, what is the index of its right child? (Your answer should be a number.)

   (e) (1 point) Consider a binary tree stored in an array as discussed in the notes and class. If a node is stored at index 7, what is the index of its parent? (Your answer should be a number.)

(f) (3 points) Consider the following binary tree:



Is it a full binary tree? ◯ Yes ◯ No
Is it a complete binary tree? ◯ Yes ◯ No
Is it a perfect binary tree? ◯ Yes ◯ No

(g) (2 points) Assume I am building a simple calculator that has the following behavior. When I see an integer, I push its value onto a stack. When I see an operator (`+, -, *, /`), I pop the stack once, and store the result in a variable called `op2`. I then pop the stack a second time, and store the result in a variable called `op1`; I then perform the arithmetic operation specified by that operator (`op1` operator `op2`) and push the result of that operation back onto the stack.

For example, if I entered `7 2 -` , `peek()` would return `5` since `7` would be pushed onto the stack, then `2`, and then the `-` would cause `2` to be popped and stored in `op2`, `7` to be popped and stored in `op1`, and the result of `7-2`, which is `5`, would be pushed onto the stack.

Tell me what value is returned by peek() after the following sequence of values is processed. Write your final answer in the box, but show you work in the empty space next to the box.

```
15 48 10 4 2 + - 3 * / +
```

$\boxed{}$

2. **Code Tracing**. Indicate what the code will print. Place your answer (and nothing else) in the box below the code.

   (a) (3 points) CT1

```java
public class CT1 {
    private class ClassA {
        protected String a;
        private int b;

        public ClassA(String a, int b) {
            this.a = a;
            this.b = b;
        }
        public void myFunc(String a) {
            this.a += a;
        }
        public String toString() {
            String ret = "";
            for (int i = 0; i < b; i++) {
                ret += a;
            }
            return ret;
        }
    }
    private class ClassB extends ClassA {
        private int c;

        public ClassB(String a, int d, int e) {
            super(a, e);
            this.c = d;
        }
        public void myFunc(String a) {
            this.a = a + this.a;
        }
    }
    public CT1() {
        ClassB b = new ClassB("ab", 2, 4);
        b.myFunc("c");
        System.out.println(b);
    }
    public static void main(String[] args) {
        new CT1();
    }
}
```

(b) (3 points) CT2

```
public class CT2 {
    Stack<Character> s = new Stack<Character>();
    ArrayList<Character> t = new ArrayList<Character>();

    public void f1(String w) {
        for (int i = 0; i < w.length(); i++) {
            s.push(w.charAt(i));
        }
    }

    public void f2(int n) {
        while (n > 0) {
            t.add(s.pop());
            n--;
        }
    }

    public void f3() {
        while (!t.isEmpty()) {
            s.push(t.remove(0));
        }
    }

    public void f4() {
        while (!s.isEmpty()) {
            System.out.print(s.pop());
        }
        System.out.println("");
    }

    public static void main(String[] args) {
        CT2 ct = new CT2();
        ct.f1("I_love");
        ct.f2(4);
        ct.f1("data");
        ct.f3();
        ct.f4();
    }
}
```

(c) (3 points) CT3

```
public class CT3 {
    public static void main(String[] args) {
        int e = 3;
        int f = 8;

        System.out.println(--e + f++ + e++ + f-- + ++e - --f);
        System.out.println(e);
        System.out.println(f);
    }
}
```

Note that there are three prints.

3. (8 points) **Big-Oh**

Determine the big-oh runtime of each of the following, in terms of N. Write your answer, and nothing else, in the box next to each method.

```java
// a contains N items, b contains 100,000 items
public static boolean big01(HashSet<Integer> a, int[] b) {
    for (int i : b) {
        if (a.contains(i)) {
            return true;
        }
    }
    return false;
}
```

```java
// a contains N items, b contains N items
public static void big02(int[] a, int[] b) {
    ArrayList<Integer> c = new ArrayList<Integer>();
    for (int i : a) {
        c.add(i);
    }
    for (int i : b) {
        if (!c.contains(i)) {
            c.add(i);
        }
    }
}
```

```
// a contains N items, b contains N items
public static void bigO3(int[] a, int[] b) {
    HashSet<Integer> c = new HashSet<Integer>();
    for (int i : a) {
        c.add(i);
    }
    for (int i : b) {
        if (!c.contains(i)) {
            c.add(i);
        }
    }
}
```

```
// a contains N items, b contains N items
public static void bigO4(int[] a, int[] b) {
    TreeSet<Integer> c = new TreeSet<Integer>();
    for (int i : a) {
        c.add(i);
    }
    for (int i : b) {
        if (!c.contains(i)) {
            c.add(i);
        }
    }
}
```

4. **Linked Lists**

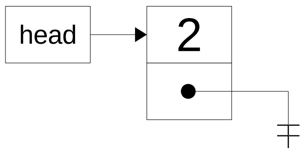
Consider the following program that creates a linked list. You may assume that the `ListNode` class exists and was defined as in class and the handout of this exam.

```
public class LinkedListCT {
    public static void main(String[] args) {
        ListNode<Integer> head = null;
        ListNode<Integer> tmp = null;

        head = new ListNode<Integer>(2);
        head.next = new ListNode<Integer>(4);
        tmp = new ListNode<Integer>(6);
        tmp.next = head.next;
        head.next = tmp;
        tmp = head;
        head = new ListNode<Integer>(8);
        head.next = tmp;
        tmp = head.next.next.next;
        head.next.next.next = null;
        tmp.next = head;
        head = tmp;
    }
}
```

Draw the state of the linked list after the execution of each specified line of code. (The linked list is defined as starting at `head`.) The first one is drawn for you.

(a) After Line 6



(b) (2 points) After Line 7

(c) (2 points) After Line 10

(d) (2 points) After Line 13

(e) (2 points) After Line 15

(f) (2 points) After Line 17

5. **Binary Search Trees (Part 1)**

   Imagine you are constructing a binary search tree of integers, and the following integers are added in the following order:

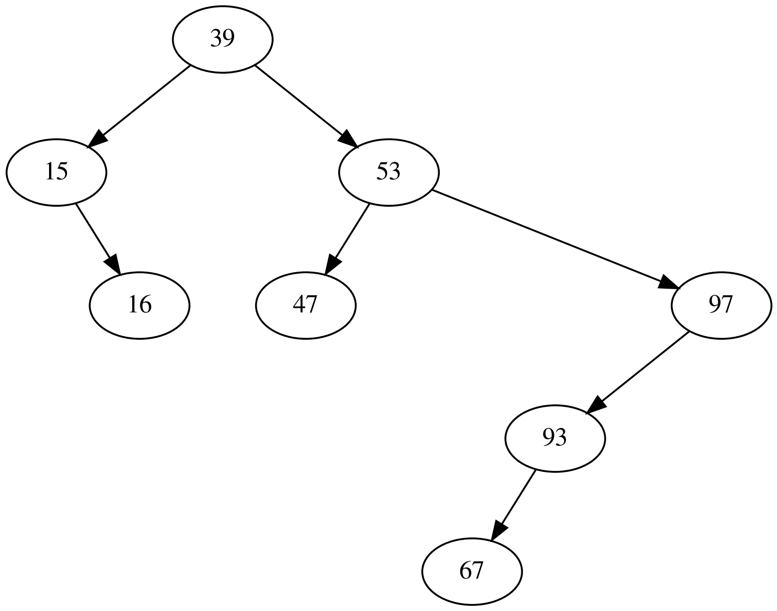
   `4, 55, 11, 82, 29, 95, 43, 27, 50, 23, 10`

   (a) (2 points) Draw the resulting binary search tree.

   (b) (2 points) Assuming you are using the in-order successor removal technique discussed in class, draw the state of your tree from (a) after 11 has been removed from it.

6. **Binary Search Trees (Part 2)**

   Consider the following tree:



   (a) (2 points) Assuming the tree is traversed *in-order* and the nodes printed, what is the resulting sequence? (Assume that left is followed before right.)

   (b) (2 points) Assuming the tree is traversed *pre-order* and the nodes printed, what is the resulting sequence? (Assume that left is followed before right.)

   (c) (2 points) Assuming the tree is traversed *post-order* and the nodes printed, what is the resulting sequence? (Assume that left is followed before right.)

7. (10 points) **Free Response**

Consider the following basic implementation of a binary search tree. All included methods, except the last one, are just like we implemented in class.

```
public class BinarySearchTree<DataType extends Comparable<DataType>> {
    private TreeNode root = null;

    private class TreeNode {
        private DataType data;
        private TreeNode left;
        private TreeNode right;

        public TreeNode(DataType data) {
            this.data = data;
        }
    }

    public void add(DataType item) {
        root = add(root, item);
    }

    private TreeNode add(TreeNode root, DataType item) {
        if (root == null) {
            return new TreeNode(item);
        }
        if (item.compareTo(root.data) < 0) {
            root.left = add(root.left, item);
        } else {
            root.right = add(root.right, item);
        }

        return root;
    }

    /**
     * Convert the tree to an ArrayList and return it. The ArrayList should be in
     * reverse sorted order (so, highest to lowest).
     *
     * @return An ArrayList containing all of the elements of the tree in reverse
     *         sorted order, or null if the tree is empty.
     */
    public ArrayList<DataType> convertToArrayList() {
        // You will write this code
    }

}
```

Write the method `convertToArrayList`. Note that you are almost certainly going to need to recursively traverse the tree in order to do this.

```
public ArrayList<DataType> convertToArrayList() {
```

8. **Free Response**

Consider the following class:

```java
public class ItemStorage<DataType> {
    private ArrayList<DataType> theGoods = new ArrayList<DataType>();

    public ItemStorage(DataType[] items) {
        for (DataType i : items) {
            theGoods.add(i);
        }
    }

    /**
     * A method to determine if there are any duplicate items in theGoods.
     *
     * This method must be O(N), where N is the size of theGoods.
     *
     * @return true if there are duplicate items and false otherwise.
     */
    public boolean containsDuplicates() {
        // You will write this code
    }

    /**
     * A method to determine if any of the items from someItems are in theGoods.
     *
     * This method must be O(N), where N is the size of both theGoods and someItems.
     *
     * @param someItems The array of items to check whether or not are in theGoods.
     * @return true if any item from someItems is in theGoods and false otherwise.
     */
    public boolean containsOneOf(DataType[] someItems) {
        // You will write this code
    }

}
```

You will write your answers on the next page.

(a) (5 points) Write the `containsDuplicates` method.

```
public boolean containsDuplicates() {
```

(b) (5 points) Write the `containsOneOf` method.

```
public boolean containsOneOf(DataType[] someItems) {
```

9. (20 points) **Free Response**

   In this problem you are going to write a `Student` class. Here are some important requirements for your class:

   - A `Student` has an age and Andrew ID. When a new `Student` is created, these items are passed to the constructor. (You must provide the constructor.)
   - The only way for code outside of `Student` to read and modify the attributes of a `Student` must be using appropriately named getters and setters. (You must provide the getters and setters.)
   - An Andrew ID must not be `null` and must be between 2 and 8 characters long. If an attempt is made to set the AndrewID to something invalid, an `IllegalArgumentException` must be thrown.
   - When printing a `Student`, its type and the value of its instance variables should appear. (Such as when `System.out.println(someStudent)` is used.)
   - A `Student` needs to be able to be properly used in `HashSet`s and `HashMap`s.
   - An array of `Student`s must be sortable using `Arrays.sort(someArrayOfStudents)`.

   Write the `Student` class to the above specifications. Pay careful attention to which classes you may need to extend or implement as well as which methods you need to write or override.

Extra space for Question 9.

More extra space for Question 9.

10. (10 points) **Free Response**

    Write the class `MinHeap` that uses an array representation to store a min-heap. (For simplicity, you must store the heap in an `ArrayList`.) Due to time restrictions of the exam, you will only be writing one public method:

    - `add(DataType item) // Add an item to the heap.`

    You may call (but must implement) any private helper methods that you need.

    The `DataType` stored by the heap should be comparable and you should use its natural ordering when determining the ordering between two items. The `add` operation must be $O(\log N)$.

    Note: Even you don't know how to write the method, properly declaring the class and including the method prototype will get you partial credit.

11. (5 points (bonus)) **Bonus Question**

    Building on your answer to Question 10, write the `removeMin` method that removes the minimum valued item from the heap.