CS151 Midterm 2

Name:

Start Time:

Finish Time:

Accommodation (if applicable):

I have abided by the Honor Code. I have not discussed this test with anyone.
(Sign below)

If you take this test on separate sheets of paper, put all of the items above on your first page.
If you need more space, feel free to add extra pages. Just make sure everything is well labelled.

There are 7 questions in this test. As indicated in the table below some questions have several parts. Be sure to answer all parts of all questions.

Question	parts	points
Name and time page		1
1	2	14
2	2	14
3	3	18
4	13	14
5	1	14
6	2	24
7	1	1
Possible Points:		100

Question 1: (14 points) The following code uses a Map to hold Stacks. Each stack contains characters. (The classes Map151 and ArrayStack are those discussed in class.)

- What is the complete contents of the variable mapp at the conclusion of the method doo? (That is show the contents of mapp as well as the contents of all things in mapp.)
- Explain (briefly) your answer.

```
   public class Q1 {
    private Map151<String, ArrayStack<Character>> mapp;

    public void doo() {
        mapp = new Map151<String, ArrayStack<Character>>();
        mapp.put("A", new ArrayStack<>(10));
        mapp.put("B", new ArrayStack<>(20));
        mapp.put("C", mapp.get("A"));
        mapp.get("A").push('e');
        mapp.get("B").push('f');
        mapp.get("C").push('g');
        mapp.get("A").push('h');
        mapp.get("B").push('i');
        mapp.get("C").push('j');
        mapp.get("A").push('k');
        mapp.get("A").pop();
        mapp.get("B").peek();

    }
    public static void main(String[] args) {
        (new Q1()).doo();
    }
}
```

```
[A:{ e g h k}]
[B:{ f i }]
[C:{ e g h k}]
```

Note that A and C are the same stack, so anything that happens to A also happens to C.

Question 2: (14 points) Using the ArrayQueue as discussing in class

- Show the exact contents of the array underlying the instance variable q2q on the completion of the do2 method.
- Explain (briefly) your answer.

```
public class Q2 {
    private ArrayQueue<Integer> q2q;

    public Q2() {
        this(8);
        q2q = new ArrayQueue<>(4);
    }

    public Q2(int siz) {
        q2q = new ArrayQueue<>(siz);
    }

    public void do2() {
        for (int i = 0; i < 10; i++) {
            if (i % 3 == 0) {
                q2q.dequeue();
            } else {
                q2q.enqueue(i);
            }
        }
        q2q.printer();
    }

    public static void main(String[] args) {
        (new Q2()).do2();
    }
}
```

```
loc     0   1    2   3
value   7   8 null   5
```

```
in the loop in do2, 1 and 2 are added to queue
in positions 0 and 1.  Then 1 is removed.  Then
4 and 5 are added in positions 2 and 3.  Then 2
is removed.   Then 7 ans 8 are added in
positions 0 and 1.  Then 4 is removed from
position 2 of array.   And we are done
```

Question 3: (18 points; 3 parts)
You are given the following array and the recursive implementation of BinarySearch shown below. Be warned, the search method does not work correctly.
Part 1 (14 points): Show the sequence of method calls, along with their returned values, that results from search(105, array) where array is as shown below

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	3	5	9	17	31	57	105	193	355	653

Part 2 (2 points): As written the search method will return the correct array location for only one value in this array. Which one?
Part 3 (2 points): Fix searchUtil

```
public class Q3 {
    public int search(int v, int[] arr) {
        return searchUtil(v, arr, 0, arr.length-1);
    }

    private int searchUtil(int tgt, int[] arr, int low, int hi) {
       if (low>hi)
          return -1;
       int mid = (low + hi) / 2;
       if (arr[mid]==tgt)
            return mid;
       if (arr[mid]<tgt)
            return searchUtil(tgt, arr, low, mid - 1);
       else
            return searchUtil(tgt, arr, mid + 1, hi);
    }
}
```

Search(105)
  searhUtil(105, arr, 0, 12)
    searchUtil(105, arr, 0, 5)
      searchUtil(105, arr, 0,1)
        searchUtil(105, arr, 0,-1)
            return -1;

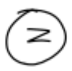
17 would be found

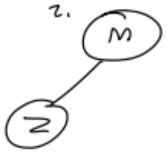
change < to > in if (arr[mid]<tgt)

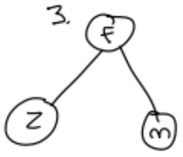
Question 4: (14 points, 13 parts) Here you will be building a heap with the SMALLEST value at the root of the heap (‘a’, is the smallest, ‘z’ is the largest) Show the contents of a heap (as a binary tree) after each of the following operations. You need not show (or explain) why the heap changed from one look to another as a result of the operation. Given that there are 13 additions or removals, there should be 13 heap tree diagrams (Grading will be 1 point per diagram. If you get one wrong, grading of subsequent diagrams will be on whether they do the correct thing from the previous diagram (if possible).)

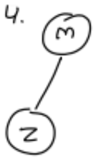
1. Add z
2. Add m
3. Add f
4. Remove smallest
5. Add e
6. Add c
7. Add a
8. Remove smallest
9. Add d
10. Add f
11. Add b
12. Add a
13. remove smallest

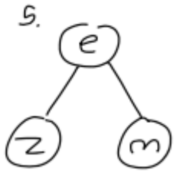
~~1.~~ Add z
~~2.~~ Add m
~~3.~~ Add f
~~4.~~ Remove smallest
~~5.~~ Add e
~~6.~~ Add c
~~7.~~ Add a
~~8.~~ Remove smallest
~~9.~~ Add d
~~10.~~ Add f
~~11.~~ Add b
~~12.~~ Add a
~~13.~~ remove smallest

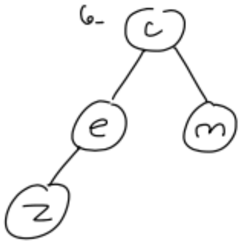
1.


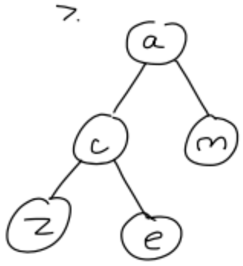
2.


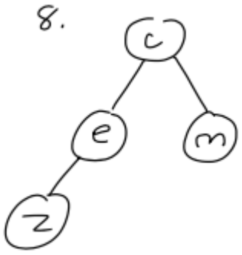
3.


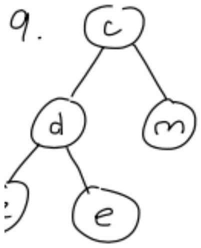
4.


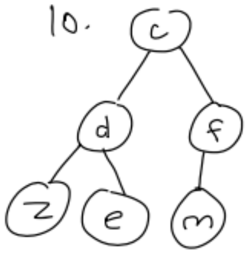
5.


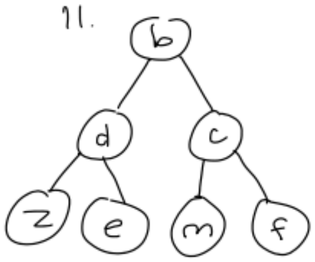
6.


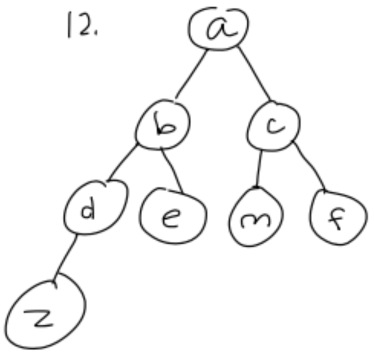
7.


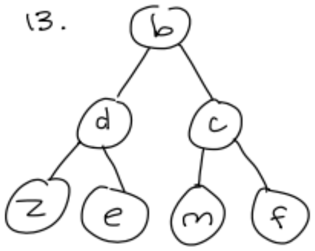
8.


9.


10.


11.


12.


13.


Question 5: (14 points) Suppose course ratings are stored in the Java class given below. You are asked to adapt this class so it implements the Comparable interface such that higher rated courses come before lower rated courses. Higher rated courses have a higher value stored in the instance variable `rating`. You recognize that there will be ties and decide on the following strategy for tie resolution. Whenever two courses have the same rating compare the courses on the basis of department name (remember String implements Comparable), except that the "CS" department should be sorted first (is the case of ties). Give the entire rewritten class.

```
public class CourseRating {
    protected int rating;
    protected String department;
    protected int courseNumber;
}
```

```
public class Q5 implements Comparable<Q5> {

    protected int rating;
    protected String department;
    protected int courseNumber;

    public Q5(int r, String d, int cn) {
        rating = r;
        department = d;
        courseNumber = cn;
    }
    @Override
    public int compareTo(Q5 o) {
        int d = rating - o.rating;
        if (d == 0) {
            d = department.compareTo(o.department);
            if (d == 0)
                return 0;
            if (department.equals("CS"))
                return 100;
            if (o.department.equals("CS"))
                return -100;
            return d;
        }
        return d;
    }
    public static void main(String[] args) {
        Q5 c1=new Q5(2,"CS",100);
        Q5 c2 = new Q5(2, "C", 100);
        System.out.println(c1.compareTo(c2));
    }
}
```

Question 6: (24 points — 2 parts) The "Fibonacci Triple" sequence is much line the standard fibonacci sequence except that instead of sequence items being based on the two previous numbers, they are based on the 3 previous numbers. Hence, the fibonacci triple sequence starts with three consecutive 1's. The fourth "fibonacci triple" number is 3 (1+1+1). The fifth is 5 (3+1+1), the sixth is 9 (5+3+1) and the seventh is 17 (9+5+3). The array in question 3 has more fibonacci triple numbers.

Part 1 (22 points): Below is a well-documented interface for the fibonacci triple. Write a class that efficiently implements this interface to compute values of the triple fibonacci sequence

Your class should only use numbers of type int; it need not deal with numeric overflow (do not use BigInteger.) You must use recursion. The methods in your class may not use any form loops (other than recursion).

Part 2 (2 points): What is the time complexity *of your implementation*.

Extra Credit: (2 points) Give a definition of "numeric overflow" and describe why it is an issue for your fibonacci triple calculator.

```
public interface FibTripInterface {
    /**
     * Compute and return the Nth fibonacci triple number.
     * the sequence returned is 1,1,1,3,5,9,17,31,57,105,...
     * So given an input of 6, this function will return 9.
     * Given 10, it will return 105.
     * @param locInSequence the position in the sequence to the
     * desired fibonacci triple number
     * @return the fibonacci triple number requested.
     * For numbers less than 0, always return 1. The returned
     * value may be incorrect when the value of n is large.
     */
    public int fibTrip(int locInSequence);

}
```

```
public class Q7 implements FibTripInterface {
    public int fibTrip(int n) {
        if (n <= 3)
            return 1;
        return fibTribUtil(1, 1, 1, n - 3);
    }

    private int fibTribUtil(int n3, int n2, int n1, int cc) {
        if (cc == 0)
            return n1;
        return fibTribUtil(n2, n1, (n3 + n2 + n1), (cc - 1));
    }
    public static void main(String[] args) {
        System.out.println((new Q7()).fibTrip(6));
    }
}
```

(page intentionally blank)

Question 7: (1 point) True/False: The founder of modern Queueing Theory worked for the AT&T at Bell Labs in New Jersey (in a building near the people who discovered Cosmic Microwave background radiation — some of the best evidence fort the Big Bang).

FALSE, her worked for the Danish Phone System.