CSC 1351: Exam 2: Midterm

VERSION A

1 Sorting

1.1

Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
public class Bug implements Comparable {
  int num_legs; // sort by this first
  int num_eyes; // sort by this if num_legs is equal
  @Override
  public int compareTo(Object o) {
    Bug that = (Bug)o;
    ..........
    int diff = this.num_legs - that.num_legs;
    ..........
    if(diff != 0) return diff;
    ..........
    return this.num_eyes - that.num_eyes;
    ..........
  }
  public static void main(String[] args) {}
}
```

1.2

What kind of sort is this? Rewrite it to use ArrayList.

```
  static void sort(int start,int[] values) {
    if(values.length-start < 2)
      return;
    int m = start;
    for(int i=start+1;i<values.length;i++)
      if(values[i] < values[m])
        m = i;
    int tmp = values[m];
    values[m] = values[start];
    values[start] = tmp;
    sort(start+1,values);
  }
```

This is a selection sort because it keeps finding the minimum and moving it to the start.

```
  static void sort(int start,ArrayList<Integer> values) {
    if(values.size()-start < 2)
      return;
    int m = start;
    for(int i=start+1;i<values.size();i++)
      if(values.get(i) < values.get(m))
        m = i;
    int tmp = values.get(m);
    values.set(m,values.get(start));
    values.set(start,tmp);
    sort(start+1,values);
  }
```

2 Anonymous Inner Classes

2.1

Given the interface defined like this:

```
public interface KeyListener {
  void keyPressed(char key);
  void keyReleased(char key);
}
```

Write a complete program that uses an anonymous inner class that implements this interface, then calls keyPressed() to print the key character to the screen.

```
public class Test {
  public static void main(String[] args) {
    KeyListener kl = new KeyListener() {
      public void keyPressed(char c) {
        System.out.println(c);
      }
      public void keyReleased(char c) {
        System.out.println(c);
      }
    };
    kl.keyPressed('A');
  }
}
```

3 Recursion

3.1

You want to implement combat within a role playing game on a computer. Specifically, the game rules for damage inflicted by a hit are:

- In order to figure out damage from one hit, you throw a N-sided die.
- The result of one throw will be between 1 and N (including both, e.g., a 6-sided die has six sides, labled 1 to 6).
- If the result is 1 to N-1, that is the resulting damage from the hit.
- If the result in N, however, you hit critically, and you throw again, adding the results.
- If you throw again, the same rules apply, potentially resulting in doubly or more critical hits.

For example, if you use a 4-sided die and throw a 3, the damage is 3. If you throw a 4 instead, you throw again. If that results in a 3, the total damage is 7. If you happen to throw two 4s after each other and then a 2, the total damage is 10. Fill in the missing code. The number of lines corresponds to the answer key. Your code may vary.

```
final static Random rand = new Random();
public static int damage(int n) {
  int result = rand.nextInt(n) + 1;
  ..........
  if (result < n) return result;
  ..........
  return result + damage(n);
  ..........
}
```

3.2

What is wrong with the following program, what happens when you run it? If this method would have been implemented using a loop, and would have a similar error, what would happen then if run?

```
public class JavaIsToJavascriptWhatCarIsToCarpet {
  public static int factorial(int n) {
    return(n * factorial(n-1));
  }
  public static void main(String args[]) {
    factorial(4);
  }
}
```

There is an exit-condition missing in the recursive function. When run, you get a stack overflow. This is similar to infinite loops, so the same in an iterative program would result in a hang during execution.

4 Big O

4.1

What is the Big-O notation for the following function?
$T(N) = 100000 + 10N + N^{-2}$
The leading term as N gets large is N, so...
$O(N)$

4.2

What is the Big O speed for a binary search? For a bubble sort? For a quick sort?
For binary search it's $O(\log N)$.
For bubble sort it's $O(N \log N)$
For quick sort it's $O(N^2)$