CS 131 Practice Final Exam
200 points

name: ________________________________

1. (6 points) What are the three main components of the stored-program architecture?

2. (5 points) Assume that `x` is a variable of type `int`. List or describe all values of `x` that will make the following Boolean expression true: `(x >= 3) || (x < 7)`

3. (6 points) Assume that `x` is a variable of type `int`. Give a Boolean expression that will evaluate to true if `x` is not between 9 and 23 inclusive.

4. (10 points) What does the term "information hiding" mean? How is information hiding implemented in Java programs?

5. Indicate the output (if any) that would be generated by the following code snippets:

   (a) (5 points)

   ```
   int x = 40;
   int y = 7;
   System.out.println(x % y);
   System.out.println(x / y);
   ```

   (b) (10 points)

   ```
   int a = 8;
   while (a >= 0) {
       if (a > 3) {
           a = a - 2;
       }
       else {
           a = a - 1;
       }
       System.out.println(a);
   }
   ```

   (c) (10 points)

   ```
   for (int i = 0; i < 3; i++) {
       String str = "";
       for (int j = 0; j <= 2; j++) {
           str = str + "*";
       }
       System.out.println(str);
   }
   ```

6. Consider the following method definition:

```
public void foo(int x, int y, int z) {
    if (x < y && y != z) {
        System.out.println("a root vegetable");
        if (y < z) {
            System.out.println("potato");
        } else if (z < x) {
            System.out.println("radish");
        } else {
            System.out.println("yam");
        }
    } else {
        System.out.println("NOT a root vegetable");
        if (y > z) {
            System.out.println("cucumber");
        } else if (y == z) {
            System.out.println("acorn squash");
        } else {
            System.out.println("tomotato");
        }
    }
}
```

(a) (5 points) What output (if any) would be produced by the following (internal) method call: `foo(3, 5, 4);`?

(b) (5 points) How many test cases would be required to achieve statement coverage for this method?

(c) (5 points) Give an example of a call to this method that could be used in a test case for covering the statement that prints “radish”.

7. (5 points) Give the base 10 value of the following binary number: 11011

8. (5 points) Give the binary representation for the following base 10 number: 21

9. (10 points) What is coupling? Give at least 2 reasons why a high degree of coupling is undesirable in a software design.

10. (10 points) If your zuul game were extended further to allow Characters to pick up and carry Items, which class should have the fields and methods for keeping track of the Items carried by a Character? Why?

11. (8 points) Consider the following snippet of code:

```
String str1 = "Final Exam";
String str2 = "final exam";
String str3 = "final exam";
String str4 = str1;
```

Indicate whether each of the following Boolean expressions would be true or false after executing this code:

(a) `str1.equals(str3)`

(b) `str3.equals(str2)`

(c) `str1 == str4`

(d) `str2 == str3`

12. (40 points) Consider the following class that keeps track of a golfer's scores for a round of golf. Each round of golf consists of 18 holes, numbered 1 through 18. Thus, the score card for a golfer will contain 18 scores, one for each hole. The score for each hole is an `int`. Complete the definition of this class by filling in the fields, constructor and the method bodies.

```
public class GolfScoreCard {
    // define fields here.







    /**
     * Construct a new GolfScoreCard for the player with the specified name.
     */
    public GolfScoreCard(String playerName) {









    }

    /**
     * Get the player's name.
     * @return the player's name.
     */
    public String getName() {





    }
```

```
/** Set the player's score for the indicated hole to the indicated score.  This
 *  method prints an error message if either the hole or the score is invalid.
 *  @param hole the hole (must be between 1 and 18 inclusive)
 *  @param score the score (must be > 0) */
public void setScore(int hole, int score) {









}

/** Get the player's total score for all 18 holes.
 *  @return the player's total score.  */
public int getTotalScore() {









}

/** Determine whether or not the player scored a hole-in-one (i.e. a score
 *  of 1 on a hole).
 *  @return the hole number on which the player scored a hole-in-one or -1
 *  if the player did not score a hole-in-one.  */
public int hasHoleInOne() {









}
```

```
    /** Return the "golf name" of the score achieved by the player on the indicated
     *  hole, which has the indicated par. The "golf names" for a score are
     *  determined as shown below:
     *
     *  Golf Name:               Description:
     *  Albatross                Score is 3 less than par.
     *  Eagle                    Score is 2 less than par.
     *  Birdie                   Score is 1 less than par.
     *  Par                      Score is equal to par.
     *  Bogie                    Score is 1 more than par.
     *  Other                    Score is greater than 1 more than par.
     *
     *  @param hole the hole number.
     *  @param par the par for the hole.  */
    public String getGolfName(int hole, int par) {













    }
}
```

13. (25 points) Consider the following class (on the next page) that represents a collection of `GolfScoreCard` objects, one for each player who is competing in a golf tournament. The number of players competing in the tournament is unknown. Complete the definition of this class by filling in the fields, constructor and the method bodies. In completing this question you should assume the existence of a correctly implemented `GolfScoreCard` class. Thus, you can receive full credit for this question even if your `GolfScoreCard` class is not completely correct.

```
public class GolfTournament {
    // Define fields here.



    /** Construct a new GolfTournament object.  */
    public GolfTournament() {



    }

    /** Add the specified GolfScoreCard to this GolfTournament.
     *  @param card a GolfScoreCard.  */
    public void addScoreCard(GolfScoreCard card) {



    }

    /** Find and return the GolfScoreCard of the winning player. The winning player
     *  is the player who has the lowest score.  In the event of a tie, this method
     *  returns the GolfScoreCard of the player most recently added to the
     *  tournament.  If there are no players in the tournament, this method prints
     *  an error message and returns null.
     *  @return the GolfScoreCard of the winner, or null if there are no players
     *         in the tournament.  */
    public GolfScoreCard findWinner() {



    }
}
```

14. (20 points) Consider the following definitions of the classes `ToDoItem` and `ToDoList`. Within these classes there are at least 6 errors that may prevent the classes from compiling, may cause the program to crash or may simply cause the program to behave incorrectly. Identify at least 5 of the errors and briefly explain why each is an error. In the interest of space, javadoc comments for the classes and methods were intentionally omitted, so you are NOT allowed to identify the lack of comments as an error.

```
public class ToDoItem {
    private String description;
    private int priority;

    public ToDoItem(String desc, int prio) {
        desc = description;
        priority = prio;
    }

    public int getPriority() {
        return prio;
    }

    public void getDescription() {
        return description;
    }
}
```

```
import java.util.ArrayList;

public class ToDoList {
    private ArrayList<ToDoItem> list;
    private String listName;

    public ToDoList(String name) {
        listName = name;
    }

    public void addItem(ToDoItem item) {
        item.add(list);
    }

    public ToDoItem getItem(int index) {
        if (index < 0 || index > list.size()) {
            System.out.println("Invalid index");
            return null;
        } else {
            return list[index];
        }
    }
}
```

15. (10 points) What output would be produced by the following snippet of code?

```
HashMap<String, String> map = new HashMap<String, String>();
map.put("toast", "breakfast");
map.put("sandwich", "lunch");
map.put("pasta", "dinner");
map.put("toast", "brunch");
map.remove("sandwich");

System.out.println(map.get("pasta"));
System.out.println(map.get("toast"));
System.out.println(map.get("eggplant"));
System.out.println(map.get("sandwich"));
```