COMP 131 Practise Final Exam (200 points total)

name: _____

Question 1. (10 points) What was the significance of the development of the stored program architecture, in which both programs and data are stored in memory?

Question 2. (10 points) Briefly explain **three** of the following four "big ideas" of software engineering: re-use, divide-and-conquer, abstraction, and modularity.

Question 3. (8 points) Fill in the body of the following method:

```
/** A very unforgiving method to evaluate students' answers
 *
 * @return true if studentAnswer is the same as
 * correctAnswer (with no difference in wording,
 * capitalization, spacing, etc.)
 */
public boolean isCorrect(String studentAnswer,
                         String correctAnswer) {









}
```

Question 4. (5 points) Assume that `x` is a variable of type int whose current value is 3. What is the value of the following expression?

```
!(x > -1) || (x == 5)
```

Question 5. (8 points) Assume that `x` is a variable of type int. Give a Boolean expression that will evaluation to true if `x` is any negative number besides -8.

Question 6.

(a) (5 points) Give the base 10 value of the following binary number: 10001

(b) (5 points) Give the binary representation for the following base 10 number: 30

Question 7. Consider the following method definition:

```
public void mystery(int x, int y, int z) {
    if (x < y || x < z) {
        System.out.println("Maybe ");
        if (y > z) {
            System.out.println("dogs");
        } else {
            System.out.println("gekkos");
        }
    } else if (y < x) {
        System.out.println("Perhaps trout");
    } else {
        System.out.println("Certainly yaks");
    }
}
```

(a) (5 points) What output would be produced by the following method call: `mystery(5,2,7)`?

(b) (10 points) Give a sequence of method calls that would produce statement coverage for this method.

Question 8. (4 points) Give the output produced by the following code snippet.

```
int a = 6;
int b = 20;
int x = b / a;
int y = b % a;
System.out.println(x);
System.out.println(y);
```

Question 9. (10 points) There is a rumor among the students at a certain college that Astronomy labs require less work than Chemistry labs. The student Senate conducts a survey of 100 students who have done each lab, collecting the total amount of hours spent throughout the semester on each lab by each student. The results can be summarized using means and standard deviations as follows:

	mean	standard deviation
Astronomy students	150	45
Chemistry students	190	30

Do the results of the survey support the hypothesis that Astronomy labs require less work than Chemistry labs, according to the criterion taught in this course? If so, how strong is this support? Justify your answer with a clear explanation and/or a diagram.

Question 10. (15 points) Explain (i) the meaning of the Java keywords `public` and `private`; (ii) how they are used to achieve information hiding; and (iii) why information hiding is desirable in software development.

Question 11. Give the output of each of the following code snippets.

(a) (10 points)

```
for (int i = 3; i < 6; i++) {
      for (int j = 5; j < 7; j++) {
            System.out.println((10 * i) + j);
      }
}
```

(b) (10 points)

```
ArrayList<String> stars = new ArrayList<String>();
int index = 0;
String theString = "";
while (index < 4) {
      theString = theString + "*";
      stars.add(theString);
      index++;
}

index = 3;
while (index > 0) {
      System.out.println(stars.get(index));
      index--;
}
```

Question 12. (10 points) Rewrite the following snippet using a `while` loop.

```
for (int i = 3; i < 9; i = i + 2) {
      System.out.println(i);
}
```

Question 13. (40 points) Fill in all the fields, constructors, and methods of the following class. Hint: only one field is required.

```
/**
 * A DailyTemperatures object stores a fixed length array of
 * temperatures, representing the daily temperature of some
 * number of consecutive days. Each day is identified with
 * an index; the earliest day has index 0, the second day
 * has index 1, and so on. All temperatures are represented
 * as integers, and are measured in degrees Fahrenheit. Note
 * that the freezing level in Fahrenheit is 32 degrees.
 */
public class DailyTemperatures {
      public static final int FREEZING = 32;
      // insert your field here



      /**
       * Construct a new object that will store the daily
       * temperatures of a fixed number of days, given by the
       * parameter numDays.
       * @param numDays the number of days whose temperatures will
       *                be stored
       */
      public DailyTemperatures(int numDays) {









      }

      /**
       * Set the value stored for the desired day to the
       * desired temperature. If the day is not valid, an
       * error message is printed.
       * @param day the index of the day whose temperature
       *            will be set
       * @param temperature the new temperature value
       */
      public void setTemperature(int day, int temperature) {









      }
```

```
/**
 * Return the temperature value stored for a given day.
 * If the day is not valid, an error message is printed
 * and the value 0 is returned.
 * @param day the index of the day whose temperature
 *            will be returned
 * @return the temperature stored at the index given by
 *         the parameter day
 */
public int getTemperature(int day) {









}

/**
 * Return the index of the first day whose temperature
 * is at or below freezing. If there are no days at or
 * below freezing, this method returns -1.
 *
 * @return index of the first freezing day
 */
public int firstFreezingDay() {









}
```

```
    /**
     * Return the number of days whose temperature is at or
     * below freezing.
     *
     * @return the number of freezing days
     */
    public int numFreezingDays() {









    }

    /**
     * A particular type of basil plant dies if it
     * encounters two consecutive days that are at or below
     * freezing. This method returns true if the
     * temperatures stored in this object would cause such a
     * basil plant to die, and false otherwise. For example,
     * the method would return true if the sequence of
     * temperatures were 45, 35, 21, 30, 42 (because the
     * third and fourth days are at or below freezing), and
     * would return false if the sequence of temperatures
     * were 45, 21, 35, 30, 42 (because no two freezing days
     * occur consecutively).
     *
     * @return true if the temperatures would kill a basil
     *         plant and false otherwise
     */
    public boolean basilPlantDies() {









    }
}
```

Question 14. (10 points) Explain why the use of the class constant `FREEZING` in the code from the previous question constitutes good software design, when compared with a design that does not use any class constants. For full credit, give at least two separate reasons with accompanying explanations. Hint: imagine the code from the previous question is being revised for use in a European country where temperature is measured in Celsius, which has a different freezing level.

Question 15. (15 points) The constructor and methods of the `CelebrityAges` class below each contain one error. Identify these 3 errors, and explain how to correct each one. (Note that the BirthYear class does not contain errors. Also, some JavaDoc comments were omitted to save space; this doesn't count as an error.)

```
/**
 * Class for storing the year in which a celebrity was born
 */
public class BirthYear {
	private int born; // year in which celebrity was born

	public BirthYear(int initBorn) {
		born = initBorn;
	}

	public int getBorn() {
		return born;
	}
}
```

```
import java.util.HashMap;
/**
 * A CelebrityAges object stores a collection of
 * celebrities' names and the years in which they were born,
 * and uses these to calculate the celebrities' ages in a
 * given year
 */
public class CelebrityAges {
      // the type of celebrity whose ages will be calculated,
      // e.g. "pop stars" or "US presidents"
      private String celebrityType;

      // A HashMap storing the names of celebrities and the
      // years in which they were born. Specifically, in each
      // key-value pair, the key is the name of a celebrity
      // (e.g. "Bill Clinton") and the value is a BirthYear
      // object which stores the year in which that celebrity
      // was born (e.g. 1946).
      private HashMap<String, BirthYear> birthYears;

      public CelebrityAges(String celebrityType) {
            this.celebrityType = celebrityType;
            birthYears = null;
      }

      public void addCelebrity(String celebrity, BirthYear birthYear) {
            birthYears.put(birthYear);
      }

      /**
       * Return the age of the given celebrity in the year
       * currentYear, or -1 if the celebrity's birth year is
       * unknown.
       */
      public int getAge(String celebrity, int currentYear) {
            BirthYear birth = birthYears.get(celebrity);
            if (birth == null) {
                  int born = birth.getBorn();
                  return currentYear - born;
            } else {
                  return -1;
            }
      }
}
```

Question 16. (10 points) The BankAccount class below is a highly simplified implementation of a bank account in which the customer's balance is stored as a whole number of dollars. Note that constructors, accessors, and JavaDoc comments have been omitted to save space. Refactor the class by factoring out the repeated code into a new method and making any necessary changes to the existing code (cross out any code you want to change and write the new code next to it.)

```
public class BankAccount {
      private int balance;

      // constructor and accessor omitted to save space

      public void printBalance() {
            String dollarWord;
            if (balance == 1) {
                  dollarWord = "dollar";
            } else {
                  dollarWord = "dollars";
            }
            System.out.println("Balance is " + balance + " "
                        + dollarWord);
      }

      public void withdraw(int withdrawalAmount) {
            if (balance - withdrawalAmount >= 0) {
                  balance = balance - withdrawalAmount;
            } else {
                  String dollarWord;
                  if (balance == 1) {
                        dollarWord = "dollar";
                  } else {
                        dollarWord = "dollars";
                  }
                  System.out.println("Error: can't withdraw, " +
                        because your balance is only " +
                        balance + " " + dollarWord);
            }
      }

      // insert new method here









}
```