1. In the Light Navigation lab, you used photo-resistor and capacitor to obtain information about light level of the room. Explain in your own words how the combination these components to accomplish this task.

2. Briefly describe what kind of code (or program) generates an ArrayOutOfBoundsException?

3. If A and B are classes and A “has a” B, what does that mean?

4. Recall the Maze Navigation lab. Assume that there can be a scenario where the MazeBot can move, but cannot reach the end of the maze. How would you modify your Maze Navigation code? Provide your answer in terms of pseudo code.

5. An object is an instance of a …?

6. Explain each of the following keyword:
   a. static
   b. this

7. Given the following Foo class:

   ```java
   public class Foo{
       private int x;

       public Foo(int x){
           this.x = x;
       }

       public int getX() { return x;}
   }
   ```

   Indicate the value of the variable `result` below:
   ```java
   Foo a = new Foo(5);
   Foo b = new Foo(5);
   boolean result = a == b;
   Answer:
   ```
   ```java
   Foo c;
   a = c;
   b = c;
   boolean result = a == b;
   Answer:
   ```
8. What is difference between return, break and continue statement?

9. Given the following array declaration:
   ```java
   int[] a = new int[5000];
   ```

   What is the type of each of the following expressions?
   a. `a.length`
   b. `a[4999]`
   c. `a`

10. What does mysteryMethod do? What does the "temp" array store? In what cases might it fail, and how could we fix this?

    ```java
    public int mysteryMethod(int[] numbers) {
        int[] temp = new int[numbers.length];
        for (int i = 0; i < numbers.length; i++) {
            temp[numbers[i]]++;
        }
        int n = 0;
        int c = temp[0];
        for (int i = 0; i < temp.length; i++) {
            if (temp[i] > c) {
                n = i;
                c = temp[i];
            }
        }
        return n;
    }
    ```
11. Consider this Toolkit class:

```java
public class Toolkit{

    public void doit1(int [] data){
        int i = 0;
        while(i < data.length){
            data[i] = data[i] * 2;
            i = i + 2;
        }
    }

    public void doit2(int [] data, int m){
        for(int i = 0; i < data.length; i++){
            if(data[i] > m){
                data[i] = m;
            }
        }
    }

}//end of class Toolkit
```

(a) What does the doit1 method do? Explain.

(b) What does the doit2 method do? Explain.
12. Consider the following code:

```java
class WeatherRecord {
    private double inchesOfRain;
    private double hiTemp;

    public WeatherRecord(double inchesOfRain, double hiTemp) {
        this.inchesOfRain = inchesOfRain;
        this.hiTemp = hiTemp;
    }

    public double getRainfall() { return inchesOfRain; }
    public double getHiTemp() { return hiTemp; }
}
```

Sample Dr Java Interactions:

```java
// Analyze the weather for a hypothetical 3-day month
WeatherRecord day1 = new WeatherRecord(0, 60); // 0 inches of rain, hi temp = 60
WeatherRecord day2 = new WeatherRecord(3, 65);
WeatherRecord day3 = new WeatherRecord(1, 68); // for simplicity, this month has 3 days
WeatherRecord[] record = new WeatherRecord[]{day1, day2, day3};
WeatherReporter reporter = new WeatherReporter();
WeatherRecord monthRecord = reporter.computeStats(record);
System.out.println(monthRecord.getRainfall()); // average rainfall for the month
Output: 1.3333333333333333
System.out.println(monthRecord.getHiTemp()); // highest temperature of the month
Output: 68.0
```

A WeatherRecord (code supplied above) is flexible in that it can hold weather information for a day, a month, even a year or a millenium.

A WeatherReporter has a single method called computeStats which takes an array of WeatherRecords as input, analyzes it, and returns a WeatherRecord which contains the results of the analysis.

Complete the computeStats method. You can assume that input array is valid (i.e. not null or length is greater than 0) and that it is full (contains no nulls).
13. Consider the following code:

```java
public class Super{
    public Super(){
        System.out.println("Super");
    }
}
```

```java
public class Sub extends Super{
    public Sub(){
        System.out.println("Sub");
    }
}
```

a. What is the output when the statement below is executed?

```java
Sub s = new Sub();
```

b. True or False: The statement below is a valid statement.

```java
Object o = new Sub();
```
14. Consider BetterBot exercise from Lab 10. Assume that you just completed writing `turnRight()` and `turnAround()` methods in the BetterBot class. Now write method called `returnToStart` which makes the BetterBot come to starting position i.e. (0, 0) from any position (x, y). As usual BetterBot inherit Bot behaviors such as `move()`, `turnLeft()`, `getDirection()`

```java
public void returnToStart()
```