1. What would happen if a wire with very little resistance was connected directly across the terminals of a 6-volt battery? Explain in your answer in terms of basic electrical circuit theory.

The low resistance of the wire would allow for a large amount of current to flow through it from one terminal of the battery to the other. The wire would get very hot due to the large current and could either fail or melt the plastic insulation.

[Figure 1]

2. Determine the four voltmeter values A, B, C and D that will register connected to the circuit as shown. Assume that battery voltage is 5V. Provide enough explanation for your answer.

A = 5V  
B = 0V  
C = 5V  
D = 0V

[Figure 2]
3. For Figure 3, the resistors that are in parallel with each other are:
   a. R2 and R4
   b. R1 and R3
   c. Both R2 and R4 along with R1 and R3
   d. R1 and (R2 + R3 + R4)
   e. None of the above

4. For Figure 3, If Req = 100 Ohms and Vs = 10V, the current leaving the voltage source must be:
   a. 1000A
   b. 1000W
   c. 0.1A
   d. 0.1W
   e. None of the above

5. To stop the servo motor on the Boe-Bot from turning, you should program the Boe-Bot to send a pulse width of 1.5 ms.

6. Assume that the BoeBot is moving through a room and there are certain instances where it must wait for 4 seconds before it can do anything. What commands will you need to perform so that BoeBot does not perform any action for 4 seconds?
   ```cpp
   for (int i = 0; i < 2; i++) {
       CPU.delay(20000);
   }
   ```

7. 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.

   Using the built-in RCTIME function, the JStamp charges up the capacitor. It then measures the time for the capacitor to discharge in the RC circuit (where R = the photoresistor). When the voltage across the capacitor drops below the threshold and the pin registers false, the time measurement stops. Depending on the relative length, the relative brightness can be ascertained.

8. 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.

In order to figure out that mazebot has come back to start we need to mark the starting position (which assume 0,0). Then its position needs update as it is moving the maze. We also need to keep track of it direction.

Note that your solution does not be as exact from the lab, for example not necessary for m.getController().moveForward(). Just an outline or pseudocode is required

Assumptions:
1. The maze is two dimensional space
2. The mazebot can determine its own NEWS and initially the default position chosen is North.
3. moveForward() is called, the move will cause position to change depending on the direction.
4. Turns are in place.
   a. For left turns it ok to just change the direction. Because once it turns left, it should still sense a wall on the right but nothing ahead and continue forward.
   b. For right turns, it important to move forward (otherwise it won't sense a wall on the right and it'll keep turning). Since we turned right and moved we must update directions and co-ordinates.

```plaintext
done = true;
x = 0; y = 0;
d = ''); //direction is initially blank
while (done) {
    if (checkRightWall && checkWallAhead()) {
        moveForward();
        if (d == 'e') {
            x = x + 1;
        } else if (d == 'w') {
            x = x - 1;
        } else if (d == 'n') {
            y = y + 1;
        } else if (d == 's') {
            y = y - 1;
        } else {
            d = 'n';
            y = y + 1;
        }
    } else if (rightWall() && checkWallAhead()) {
        turnLeft();
        if (d == 'e') {
            d = 'n';
        } else if (d == 'w') {
            d = 's';
        } else if (d == 'n') {
            d = 'w';
        } else if (d == 's') {
            d = 'e';
        } else {
            d = 'n';
            y = y + 1;
        }
    } else {
        turnRight();
        moveForward();
        if (d == 'e') {
            d = 'n';
            x = x + 1;
        } else if (d == 's') {
            d = 'w';
            x = x - 1;
        } else if (d == 'n') {
            d = 's';
            y = y + 1;
        } else if (d == 'w') {
            d = 'e';
            y = y - 1;
        } else {
            d = 'n';
        }
    }
    if (x == 0 && y == 0) {
        done = false;
    }
}  
}  
}  
end while

stop();  //stop moving
```
9. Infer the type of variable x:
   \[ x = !x \ &\ & (x == \text{false}); \]
   \textbf{boolean}

10. What is the value of variable x after the following code snippet is executed:

\begin{verbatim}
int x = 4;
if(x > 0)
    x = x + 10;
    x = x - 5;
if(x <= 0)
    x = 50;
else
    x = x * 10;
\end{verbatim}

\textbf{x = 90}. Note that statement 2 i.e. \( x = x - 5 \) is not part of the if \((x > 0)\). If we do not write curly brace, then only next statement after if statement is part of the if condition. If you thought that both statement 1 and statement 2 were part of if the block, then try \( x = 0 \) and see what you get.

11. Using parentheses, show how Java will interpret the expressions below:
   a. \( 2 / 3 \ % 4 + 8 - ((2 / 3) \ % 4) + 8) \)
   b. \( \text{true} \ &\& \text{true} || \text{false} - ((\text{true} \ &\& \text{true}) \ || \text{false}) \)

12. An object is an instance of a … ? \textbf{class}

13. Explain each of the following keyword:
   a. static - \textit{variable/method} declared static belongs to the class, not to \textit{individual objects}
   b. this - \textit{the object that is executing the current code} (value of this is the object's heap address)

14. Given the following array declaration:
   \begin{verbatim}
   int [] a = new int [5000];
   \end{verbatim}

What is the type of each of the following expressions?
   a. \texttt{a.length} - \textbf{int}
   b. \texttt{a[4999]} - \textbf{int}
   c. \texttt{a} - \textbf{int []}
15. 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;
}
```

temp stores a count of each item in the array. If 2 shows up 3 times, then temp[2] = 3. The second loop finds the maximum frequency and returns the value. So if 7 appeared most frequently in the input array, it returns 7. This method can fail due to indexOutOfBoundsException for line that does the statement temp[numbers[i]]++. This can be fixed with an if-statement check.

16. 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:

Foo a = new Foo(5);
Foo b = new Foo(5);
boolean result = a == b;
Answer: false

Foo c;
a = c;
b = c;
boolean result = a == b;
Answer: true
17. Consider the following code:

```java
public 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).
public class WeatherReporter{

    //Method computeStats

    public WeatherRecord computeStats(WeatherRecord[] record){
        double monthRain = 0;
        for (int i = 0; i < records.length; i++){
            monthRain += records[i].getRainfall();
        }
        double avgMonthRain = monthRain/records.length;
        double monthHiTemp = records[0].getHiTemp();
        for (int i = 1; i < records.length; i++){
            if (records[i].getHiTemp() > monthHiTemp){
                monthHiTemp = records[i].getHiTemp();
            }
        }
        return new WeatherRecord(avgMonthRain, monthHiTemp);
    }
}

18. 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()

public void returnToStart(){
    if(getDirection() == 's')
        turnAround();
    else if(getDirection() == 'w')
        turnRight();
    else if(getDirection() == 'e')
        turnLeft();

    while(move());
    turnLeft();
    while(move());
    turnAround();
}