Name ____ Period ________

Blob Hunter

Your Tasks (Mark these off as you go)

- ☐ Practice with recursion
- ☐ Write recursive algorithms
- ☐ Write the getBlob method for one-dimension
- ☐ Write the getBlob method for two-dimensions
- ☐ Receive credit for this lab guide

☐ Practice with recursion

Trace the following code segments on the paper provided. Indicate the stack and the output for each. Write the result your group agrees upon below.

Code	Stack	Output

`public static void showMe(int arg)`
`{`
`      if (arg < 10)`
`      {`
`            showMe(arg + 1);`
`      }`
`      else`
`      {`
`            System.out.print(arg + " ");`
`      }`
`}`
 	
`public static void main(String args[]){`
 	
`      showMe(0)`
`}`		

Code	Stack	Output

```
public static void whatsItDo(String str)
{
   int len = str.length();
      if (len > 1)
      {
 String temp = str.substring(0, len – 1);
 System.out.println(temp);
 whatsItDo(temp);
      }
}

public static void main(String args[]){

     whatsItDo("WATCH")
}
```

Code	Stack	Output

```
public static void puf(int n)
{
     if(n == 1)
     {
          System.out.print(“x”);
     }
     else if( n%2 = = 0) //n is even
     {
          System.out.print(“{”);
          puf(n-1);
          System.out.print(“}”);
     }
     else //n is odd
     {
          System.out.print(“<”);
          puf(n-1);
          System.out.print(“>”);
     }
}

public static void main(String args[]){

     puf(5);
}
```

Code	Stack	Output

`public int addFun(int n)`
`{`
`   if (n <= 0)`
`      return 0;`
`   if (n == 1)`
`      return 2;`
`   return addFun(n - 1) + addFun(n - 2);`
`}`
 	
`public static void main(String args[]){`
 	
`     System.out.println(addFun(6));`
`}`		

□ Write recursive algorithms

Many algorithms that incorporate loops can also be written recursively. Consider the following methods which reverse a number and return the value.

Number reversal using a loop

```
public static int reverseNum(int n, int digits){
        while(n > 0){
            reverse += n%10 * Math.pow(10, digits);
            digits--;
            n /= 10;
        }
        return reverse;
}
```

Number reversal using recursion

```
public static int reverseNumRecursion(int n, int digits){
        if(n == 0){
            return reverse;
        }else{
            return (int)(n%10 * Math.pow(10, digits)) +
reverseNumRecursion(n/10, digits - 1);
        }
}
```

The countFlips method is intended to count the number of flips it takes to achieve a certain number of heads in a row. Chat GPT was asked to produce an algorithm that does this recursively and generated the code below.

```java
public static int countFlips(int headsInARow) {
        // Base case: If required number of heads reached, return 0 flips needed
        if (headsInARow == 0) {
            return 0;
        }

        if (Math.random() < 0.5) {
            return 1 + countFlips(headsInARow - 1);
        } else {
            return 1 + countFlips(headsInARow);
        }
}
```

In the space below, explain (using actual words not code, why this code fails). It may be helpful to run the code in your editor and run some test cases. Then using words (not code), explain what you would do to fix it. Use complete coherent sentences.

The following method converts a base10 number to binary.

```java
public static long decToBin(int decimal){
int count = 0;
long output = 0;
        while(decimal > 0){
            output += (long)((decimal%2)*Math.pow(10, count));
            decimal = decimal/2;
            count++;
        }
return output;
}
```

In the space below, re-write the code as a recursive method (Don't resort to cheating, play with the code to find a valid solution)

☐ Write the getBlob Method for one-dimension

The getBlob() method recursively searches for areas on a grid that do not contain mines. For example, if a user clicks on index 2 in the 1-dimensional grid below, buttons 2 and 3 will change color as shown.

If a user clicks on indices 1 or 4 however, they lose.

Likewise, if a user clicks on index 6, buttons 5, 6, and 7 will change color.

0	1	2	3	4	5	6	7	8	9
	M	User clicked	▨	M	▨	User clicked	▨	M	

To get started on this method, think about the base cases:

- If a mine is found
- If a user goes out of bounds
- If a button is visited

In other words, if a user clicks on a button that does not contain a mine and is within the boundaries of the grid and has not been visited, we can call the recursive method, otherwise we will not.

Another way to state this, is as follows,

```
if (b >= 0 && b < gridDimensions && mines[b] == false && visited[b] == false) {
    //color buttons
    //set buttons to visited = true
     //run the recursion portion
}
```

Write the getBlob method for one dimension. Your method should accept one parameter with represents the x location (or index) of the button clicked.

In the body of the if statement,

- set the visited location to true
- call getBlob for each adjacent button
- Set the background of all buttons in the block to orange - tiles[b].setBackground(Color.orange);

☐ Write the getBlob method for two-dimensions

Now that you have figured out the logic for a one-dimensional getBlob method, you can expand it to two dimensions. To do this you will need to think about the additional boundary cases. You will also need to think about the additional recursive calls.

Consider the following two-dimensional grid. If a user clicks on the locations shown, the buttons should change colors as shown. If a user clicks on a mine (M), the user will lose.

	M	M	▨
M	▨ User Click	M	▨ User Clicked
▨	▨	M	▨

The signature for the getBlob method for two-dimensions is below. Write the getBlob method for two-dimensions.

```java
public void getBlob(int r, int c)
```

☐ Receive Credit for this lab guide

Submit this portion of the lab to Pluska to receive credit for the lab guide. Once received, your completed code challenges will also be graded and will count towards your final lab grade.