1) **Part 1: Given the following code draw the GUI, Part One is when the program starts up before user interaction. Draw out your answer, with small text label the items with p1, p2, p3 and their approximate size, and put the buttons with text where they belong. Then fill out the three questions for part one explaining their colors. Then do part two.**

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class SwingQuestion extends JFrame
{
   JPanel panel, p1, p2, p3;
   JButton b1, b2;

   SwingQuestion()
   {
     super("Tough Question");
     setSize(400, 400);
     panel = new JPanel();
     p1 = new JPanel();
     p2 = getPanel();
     p3 = new JPanel();
     p1.setBackground(Color.red);
     p3.setBackground(Color.blue);
     p1.addMouseListener(new MyStuff());
```

```
    panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
    panel.add(p1);
    panel.add(p2);
    panel.add(p3);
    getContentPane().add(panel);
    setVisible(true);
  }
   public JPanel getPanel()
  {
     JPanel temp = new JPanel();
     temp.setBackground(Color.white);
     b1 = new JButton("start");
```

/*Code continued on next page*/

```
     b1.addActionListener(new MyStuff());
     b2 = new JButton("stop");
     b2.addActionListener(new MyStuff());
     temp.add(b1);
     temp.add(b2);
     return temp;
  }

  public class MyStuff implements MouseListener, ActionListener
  {
     public void actionPerformed(ActionEvent evt)
```

```
  {
     String command = evt.getActionCommand();
     if(command.equals("start"))
     {
       p3.setBackground(Color.yellow);
     }
     else if(command.equals("stop"))
     {
        p2.setBackground(Color.gray);
     }
     else if(command.equals("b1"))
     {
        p3.setBackground(Color.black);
     }
     else
     {
        p1.setBackground(Color.black);
     }
  }
  public void mousePressed(MouseEvent evt){}
  public void mouseReleased(MouseEvent evt){b2.setText("Over");}
  public void mouseClicked(MouseEvent evt){}
  public void mouseEntered(MouseEvent evt){b1.setText("Done");}
  public void mouseExited(MouseEvent evt){}
}
```

```
    public static void main(String [] args)
    {
        new SwingQuestion();
    }
}
```

Draw your frame (use lines to show borders of panels):

Question one -Part 2: After the program starts the user completes the steps listed at the given locations. After all moves, mouse clicks and button pushes are completed give the color of the panels and the text on the buttons.

A. Mouse starts in the very middle of the screen.
B. Moves to the button on the (<-)LEFT side of the screen and clicks
C. Moves to the button on the (->) RIGHT side of the screen and clicks
D. Moves to the top part of the Frame, still on a panel and clicks the mouse
E. Goes back to the button on the far right (->) and clicks, same button as step C.
F. Goes back to the button on the far left (<-) and clicks, same button as step B.
G. Moves to the bottom part of the frame, still on a panel and mouse clicks.

Part 2 Answers:

Color of p1:

Color of p2:

Color of p3:

Text on b1:

Text on b2:

```
int factorial(int n) {
   if (n <= 1)
      return 1;
   else
      return (n * factorial(n-1));
}


int sum1toN(int n) {
   if (n < 1)
      return 0;
   else
      return (n + sum1toN(n-1));
}
```

1. Identify the termination conditions (and resulting base case return values) in each of the above recursive methods.

   (n<=1) and (n<1)

2. Identify the recursive calls in each of the above methods.

   `return n * factorial(n-1));`

   `return (n + sum1toN(n-1));`

3. Using this call `factorial(5)`, draw the execution stack as it would look just before returning from `factorial(1)`, the last recursive call.

   Stack like I drew on the board several times.

4. The data flow model is yet another way to illustrate a recursive computation. In the data flow model, you use boxes to represent method calls and arrows to represent the flow of information (parameters and return values). You can draw each recursive call as a black box with the parameters feeding in at the top left, and the return value coming out at the bottom left. To illustrate a method call, the parameters go out of the caller's right side and into the left side of the method being called. Return values go the other direction.

   Just like part 3 above, just a different look.

5. Write a recursive method that takes as parameters an initial investment amount, an annual interest rate, and a number of years. The method should return the value of the investment after the given number of years, assuming that the interest is compounded annually. (For example, if the initial investment is 1000 and the interest rate is 10 percent, then after one year the investment will be worth 1100, after two years 1210, after three years 1331, etc.)

```
public static double recur(double amt, double rate, int year)
   {
      if(year < 1)
        return amt;
      else

        return recur(amt * (1 + rate), rate, year-1);
      }
```

Reference Question (14 points) The main method is on the next page in the **Ref** class, this page has the **Toons** class. On the next page give the output of the seven print statements. No errors are included it compiles and works just fine.

```
public class Toons
{
	public static int num = 0;
	private String feature;

	public Toons()
	{
		feature = "Looney";
		num+=6;
	}

	public Toons(String in_feature)
	{
		feature = in_feature;
		num += 3;
	}

	public Toons(String in_feature, int a)
	{
		feature = in_feature;
		num = num + (a*2);
	}

	public String getFeature()
	{
		return feature;
	}

	public void setFeature(String haha)
	{
		feature = haha;
	}

	public Toons testQuestion(Toons sat, Toons sun, Toons mon)
	{
		sat.setFeature("Elmer");
		sun.setFeature(mon.getFeature());
		mon = this;
		System.out.println(sat.getFeature());	//Statement 1
		System.out.println(sun.getFeature());	// Statement 2
		System.out.println(mon.getFeature());	// Statement 3
		return sun;
	}
}
```

```
public class Ref

{
        public static void main(String [] args)
        {
                Toons cartoon = new Toons("Bugs", 6);
                Toons nostril = new Toons("Blanc");
                Toons  moon = cartoon;

                quack = cartoon.testQuestion(beak, nostril, new Toons("Mel", 2));

                System.out.println(cartoon.getFeature());               // Statement 4
                System.out.println(nostril.getFeature());       // Statement 5
                System.out.println(moon.getFeature());          // Statement 6
                System.out.println(moon.num);                           // Statement 7
        }

}
```

Print Statement 1:

Print Statement 2:

Print Statement 3:

Print Statement 4:

Print Statement 5:

Print Statement 6:

Print Statement 7:

```
Elmer
Mel
Elmer
Elmer
Mel
Elmer
```

Given the language:

L = {w$w' : w is possibly empty string of characters other than $,

and w' = reverse(w)}

write an explanation (a little psuedocode method) of how you would use a stack to solve whether any given string is in this language.

Put letters in the Strig on the stack until you get to the $ then pop off matching the next letter in the String after the $ sign with the top letter on the stack.

QUESTION Five- Reference Question (14 points) The main method is on the next page in the **Ref** class, this page has the **Toons** class. On the next page give the output of the seven print statements. No errors are included it compiles and works just fine.

```
public class Toons
{
	public static int num = 0;
	private String feature;

	public Toons()
	{
		feature = "Looney";
		num+=6;
	}

	public Toons(String in_feature)
	{
		feature = in_feature;
		num += 3;
	}

	public Toons(String in_feature, int a)
	{
		feature = in_feature;
		num = num + (a*2);
	}

	public String getFeature()
	{
		return feature;
	}

	public void setFeature(String haha)
	{
		feature = haha;
	}

	public Toons testQuestion(Toons sat, Toons sun, Toons mon)
	{
		sat.setFeature("Elmer");
		sun.setFeature(mon.getFeature());
		mon = this;
		System.out.println(sat.getFeature());      //Statement 1
		System.out.println(sun.getFeature());     // Statement 2
		System.out.println(mon.getFeature());   // Statement 3
		return sun;
	}
}
```

```
public class Ref

{
        public static void main(String [] args)
        {
                Toons cartoon = new Toons("Bugs", 6);
                Toons nostril = new Toons("Blanc");
                Toons  moon = cartoon;

                quack = cartoon.testQuestion(beak, nostril, new Toons("Mel", 2));

                System.out.println(cartoon.getFeature());               // Statement 4
                System.out.println(nostril.getFeature());       // Statement 5
                System.out.println(moon.getFeature());          // Statement 6
                System.out.println(moon.num);                           // Statement 7
        }

}
```

Print Statement 1:

Print Statement 2:

Print Statement 3:

Print Statement 4:

Print Statement 5:

Print Statement 6:

Print Statement 7:

```
Elmer
Mel
Elmer
Elmer
Mel
Elmer
```