Unit 1 – ICS4U0 – Graphics & Applets

Sample Test – Friday September 23, 2022

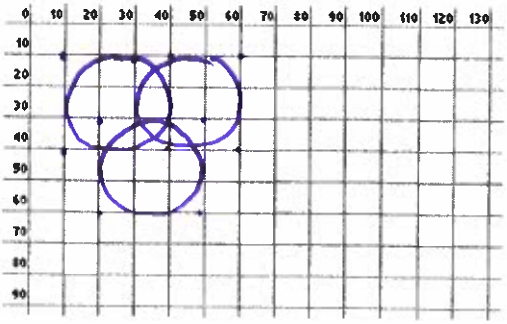
Name: Solutions

Total	%	Knowledge	Thinking	Communication	Application
(94)	%	(23)	(23)	(22)	(26)

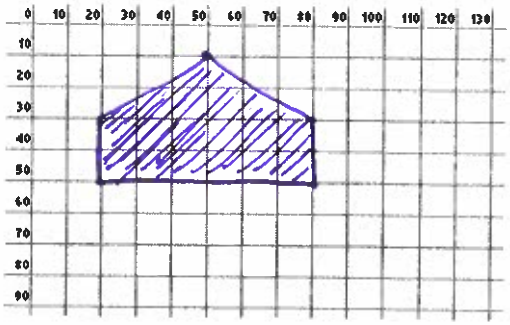
Knowledge

1. Draw what is drawn by this code. /8

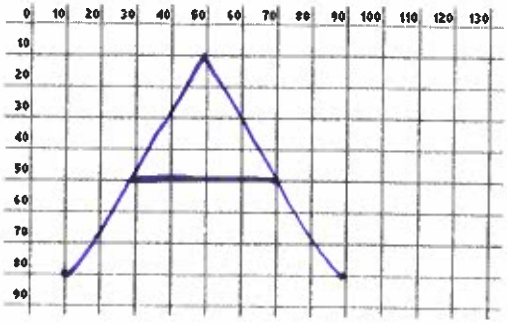
```
import java.applet.*; import java.awt.*;
public class partA extends Applet {
    public void paint (Graphics g) {
        g.drawOval (10, 10, 30, 30);
        g.drawOval (30, 10, 30, 30);
        g.drawOval (20, 30, 30, 30);
    }
}
```



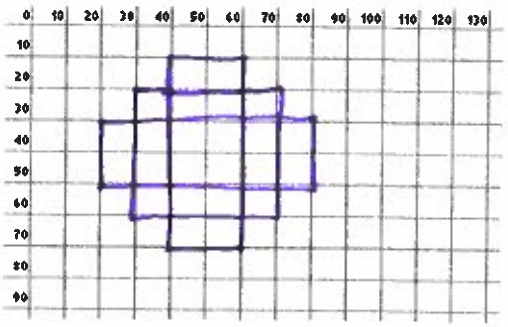
```
import java.applet.*; import java.awt.*;
public class partB extends Applet {
    public void paint (Graphics g) {
        g.setColor (Color.blue);
        int x [] = {50, 80, 80, 20, 20};
        int y [] = {10, 30, 50, 50, 30};
        g.fillPolygon (x, y, 5);
    }
}
```



```
import java.applet.*; import java.awt.*;
public class partC extends Applet {
    public void paint (Graphics g) {
        g.setColor (Color.green);
        g.drawLine (50, 10, 10, 80);
        g.drawLine (30, 50, 70, 50);
        g.drawLine (50, 10, 90, 80);
    }
}
```



```
import java.applet.*; import java.awt.*;
public class partD extends Applet {
    public void paint (Graphics g)  {
        g.setColor (Color.red);
        g.drawRect (40, 10, 20, 60);
        g.drawRect (30, 20, 40, 40);
        g.drawRect (20, 30, 60, 20);
    }
}
```

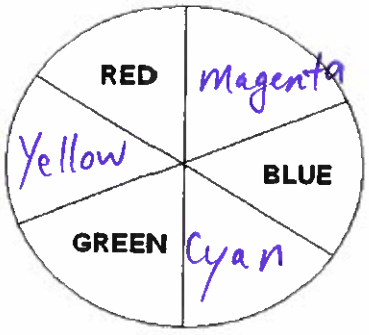


2. Write the code required for each colour. /6

R G B

black	title.setBackground(new Color(0 , 0 , 0));
white	title.setBackground(new Color(255,255,255));
green	title.setBackground(new Color(0 ,255, 0));
cyan	title.setBackground(new Color(0 ,255,255));
blue	title.setBackground(new Color(0 , 0 ,255));
magenta	title.setBackground(new Color(255, 0 ,255));
red	title.setBackground(new Color(255, 0 , 0));
yellow	title.setBackground(new Color(255,255, 0));

3. Fill in this colour wheel based on the computer colours. Use the proper colour names. /3



4. Fill in the code that would be used in the init method to make the widgets shown. /6

(a)

white writing on black
regular font
ActionCommand: layers

```
JButton tf = new JButton ("Onion");
tf.setActionCommand ("layers");
tf.addActionListener (this);
tf.setForeground (Color.white);
tf.setBackground (Color.black);
add (tf);
```

(b)

Picture: sun.jpg
ActionCommand: yellow

```
JButton sun = new JButton (createImageIcon("sun.jpg"));
sun.setActionCommand ("yellow");
sun.addActionListener (this);
add (sun);
```

(c) Fill in actionPerformed with the appropriate information from parts (a) and (b).

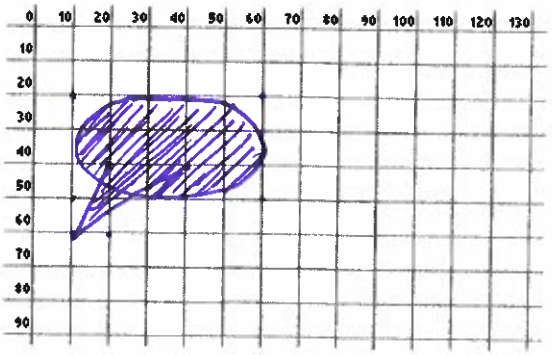
```
public void actionPerformed (ActionEvent e) {
   if (e.getActionCommand ().equals ("layer")) {
      showStatus ("The Onion button was pressed");
   }
   else if (e.getActionCommand ().equals ("yellow")) {
      showStatus ("The Sun button was pressed");
   }
}
```

Thinking

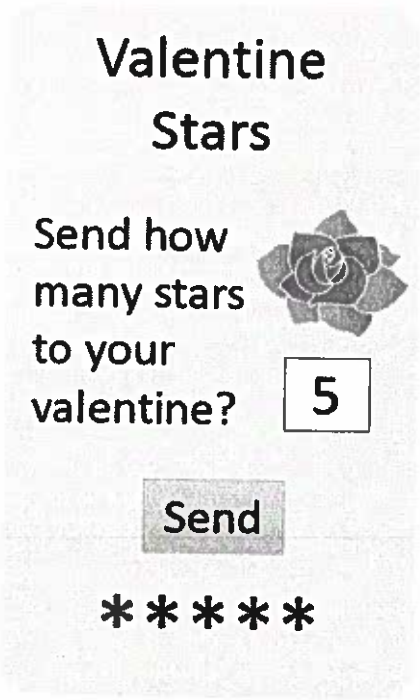
5. Draw what is drawn by this code. Do **not** assume that the shape is drawn at 0,0 /4

```
import java.awt.*; import java.applet.Applet;
public class sampleTest extends Applet
{
  public void paint (Graphics g)
  {   speechBubble (10, 20, 0, 0, 255);
  }

  public void speechBubble (int x, int y, int r, int gg, int b)
  {   Graphics g = getGraphics ();
      g.setColor (new Color (r, gg, b));
      g.fillOval (x, y, 50, 30);
      int xs[] = {x + 10, x + 30, x};
      int ys[] = {y + 20, y + 20, y + 40};
      g.fillPolygon (xs, ys, 3);
  }
}
```



6. White Box and Black Box test this app. /10

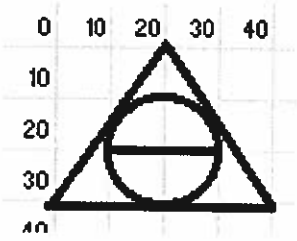


Black Box Test Case	Black Box Data
short data	2
long data	7
average data	5
boundary data	-1, 0, 1
White Box Test Case & line number	**White Box Data**
4 - avoid loop	-2 negative, 0
4 - run loop once	1 1
4 - run loop many times	8 over 1
6 - if (n<0)	-1 0 or under
8 - else if (n>8)	10 over 8
10 - else	6 1-8

The Code:

```
1  public void actionPerformed(ActionEvent e){
2  int n = Integer.parseInt(howmany.getText());
3  String ans = "";
4  for(int i=0; i<n; i++)
5          ans+="*";
6      if(n<=0)
7          answer.setText("Too small");
8      else if(n>8)
9          answer.setText("Too many");
10     else
11         answer.setText(""+ans);
12 }
```

7. Circle and correct one error on each line of code. /9




```
public void wackyTriangle (int x, int y)
{
    Graphics bob = getGraphics ();
    int xs[] = {x + 20, x + 40, y};
    int ys[] = {0, y + 30, y + 30};
    g.drawPolygon (xs, ys, 19);
    g.drawOval (x + 10,y + 10, 20, 20);
    g.drawLine (x + 10, y + 20, 30, 20);
}
```

Communication

8. Provide the phrase or word required. /12

drawRect	(a)	The graphics method needed to draw a square.
drawPolygon	(b)	A graphics method with two arrays and an int as parameters.
Ariane 5	(c)	Blew up due to poor testing. Cost $7.5 billion.
Cancer	(d)	The disease treated by the American Megatrends device. – in Panama
drawPolygon	(e)	The graphics method needed to draw a triangle in java.
parameters	(f)	Input coming into a method.
paint	(g)	The "main method", or starting point, in a graphics Applet.
code (aka create)	(h)	The phase of the PDLC with testing.
method signature	(i)	The name for the first line of a method.
method	(j)	A subprogram or function in java.
abstraction	(k)	The 'A' in ORATE stands for this method principle.
extensibility	(l)	The 'E' in ORATE stands for this method principle.

O - organization
R - reusability
A - abstraction
T - testing
E - extensibility

9. Choose true or false for the following statements. They are based on the code to the right. /5

(T) F (a) The method name is cloud.

T (F) (b) There are 3 parameters. There are 2

T (F) (c) There are 2 method signatures. only 1 per method

T (F) (d) The return type is int. void

T (F) (e) The cloud is drawn in blue. magenta

```
public void cloud (int x, int y) {
  Graphics g = getGraphics ();
  g.setColor (new Color(255,0,255));(R) G (B)
  g.fillOval (x + 20, y, 30, 30);
  g.fillOval (x + 40, y + 10, 30, 20);
  g.fillOval (x + 20, y + 20, 30, 30);
  g.fillOval (x + 40, y + 20, 30, 30);
  g.fillOval (x + 10, y + 30, 20, 20);
  g.fillOval (x, y + 10, 30, 30);
}
```

magenta

10. Explain how graphics methods help with organization of the code. /2

Organization refers to how methods break things up into smaller logical units. In the cloud method above, the logical unit is the code required to make a cloud shape. This is all neatly packaged in the { } and is titled "cloud" so it is easy to find & use.

An specific example makes a great second point.

11. How would you test a new elevator? [FYI: this is a google interview question] /3

BlackBox - Average - Put a series of weights on the elevator to represent an average number of people.

Use terminology from class.

BlackBox - Boundary - Test what happens when the power goes out

BlackBox - Large - Fill the elevator to capacity for a long period of time. Have it stop frequently.

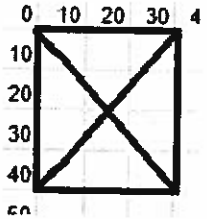
WhiteBox - Test all ifs - Test each of the buttons on each floor.

White Box - Run loop many times - Run elevator constantly for weeks.

Application

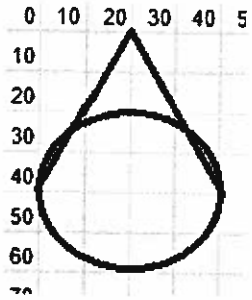
12. Code each method so that it is moveable on the screen. /8

```
public void xMarksTheSpot (int x, int y) {
    Graphics g = getGraphics ();
    g.drawRect ( x , y , 30, 40 );
    g.drawLine ( x, y , x+30, y+40 );
    g.drawLine ( x+30, y , x, y+40 );
}
```

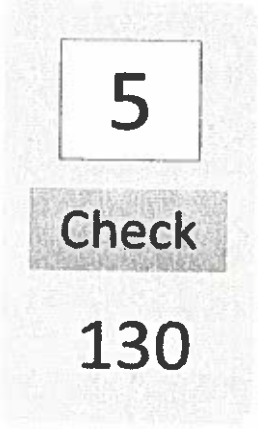



```
public void rainDrop (int x, int y, int r, int gg, int b) {
    Graphics g = getGraphics ();
    g.setColor (new Color (r, gg, b));
    g.fillOval ( x , y+20 , 40, 40 );
    g.setColor (Color.white);
    g.fillRect (                                    );
    g.setColor (new Color (r, gg, b));
    int xs[] = { x+20, x+40 , x };
    int ys[] = { y , y+40 , y+40 };
    g.fillPolygon (xs, ys, 3 );
}
```

sigh.



13. A number of test cases were run on this app and the results of each test are shown in the table below. What is the code in the actionPerformed method named? (The JTextField is `input`; the JLabel is `output`) /8



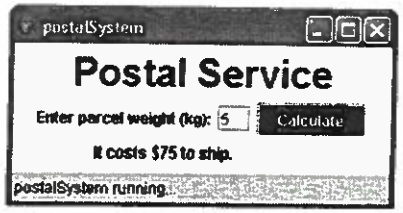

```
public void actionPerformed (ActionEvent e)
{
    int i = Integer.parseInt (input.getText());
    int math = i * i * i + i ;
    output.setText ("" + math);
}
```

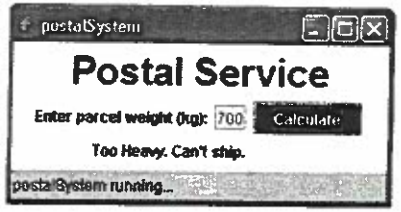
Test Case	Output
4	68
1	2
120	1728120
0	0
-135	-2460510
2	10

14. Write the code for the init method of this program. /7

```
import java.awt.*; import javax.swing.*; import java.applet.Applet; import java.awt.event.*;
public class postalSystem extends Applet implements ActionListener
{
    JTextField weight;
//The other global variable:
    JLabel cost;
    public void init ()
    {  resize (300, 100);
//The first label: (Font is Arial, Font.BOLD and 30 pt)
    JLabel title = new JLabel ("Postal Service");
    title.setFont(new Font("Arial",
         Font.BOLD, 30));
//The instruction
    JLabel ins = new JLabel ("Enter parcel weight(kg)");
//The textfield:
    weight = new JTextField (5);
//The button: (Don't forget the colour changes!)
    JButton b = new JButton ("Calculate");
    b.setBackground(Color.black);
    b.setForeground(Color.white);
    b.addActionListener(this);
    b.setActionCommand("Calculate");
//The last label:
    cost = new JLabel ("Cost here");
//add the widgets:
    add(title);
    add(ins);
    add(weight);
    add(b);
    add(cost);
} //init
```

15. Write the code for the postalSystem actionPerformed method.





The calculations used by the program are below. /3

Weight Range	*Message*
Over 500 kg	Too Heavy. Can't ship.
Between 50 and 500kg	It costs $150 to ship
Between 2 and 50 kg	It costs $75 to ship.
Under 2 kg	Too Light. Can't ship.

```
public void ActionPerformed (ActionEvent e)
{
  if (e.getActionCommand ().equals ("calculate"))
  {
    int w = Integer.parseInt (weight.getText ());
    if (w > 500)
       cost.setText ("Too Heavy. Can't ship.");
    else if (w > 50)
       cost.setText ("It costs $150 to ship.");
    else if (w > 2)
       cost.setText ("It costs $75 to ship.");
    else
       cost.setText ("Too Light. Can't ship.");
  }
```

"green" parts can be different, but need to match.