Family Name: ..... **Other Names:** .....

Student ID: ..... **Signature** .....

COMP 102/112 : Test 1

2023, April 27 ** WITH SOLUTIONS **

Instructions

- Time allowed: **60 minutes**
- Attempt **all** the questions. There are 50 marks in total.
- Write your answers in this test paper and hand in all sheets. (We have different instructions for distance students)
- If you think a question is unclear, ask for clarification.
- Brief Java documentation is provided with the test.
- This test contributes 30% of your final grade.
  (But your mark will be increased to your exam mark if that is higher.)
- You may use dictionaries and calculators.
- You may write notes and working on this paper, but make sure your answers are clear.
- You may assume all the programs import the ecs100 library and other standard libraries.

Questions	**Marks**	
1. User input	[10]	
2. Writing programs with if	[10]	
3. Defining methods with parameters	[10]	
4. Writing methods that use objects	[10]	
5. Writing programs with for	[10]	
	TOTAL:	

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 1. User Input **[10 marks]**

Complete the printAverage() method below so that it:

- Asks the user for two doubles.
- Prints out the average of the two values.

For example:

```
First number:  5
Second number:  10
The average is 7.5
```

Remember: Given two numbers n1 and n2, the average is (n1 + n2)/2

```
public void printAverage () {



    double n1 = UI.askDouble("First number:");
    double n2 = UI.askDouble("Second number:");

    double average = (n1 + n2)/2;

    UI.println ("The average is " + average);
//OR
    UI.printf ("The average is %.1f\n", average);









}
```

Question 2. Writing programs with if **[10 marks]**

Your task is to complete the findPostage(...) method to **return** a String message for the number specified in the parameter, according to the following table.

Weight	Message
0 or below	Error
above 0 up to 10	$3.00
from 10 up to 200	$3.80
other values	Not a letter

For example, if findPostage(...) is called with the below arguments, it returns:

argument	returns
-1	Error
0	Error
2	$3.00
10	$3.80
150	$3.80
200	Not a letter

```
/** Returns the postage associated with the given weight */
public String findPostage(double wgt){




    if (wgt<= 0)
        return "Error";
    else if (wgt<10)
        return "$3.00";
    else if (wgt<200)
        return "$3.80";
    else
        return "Not a letter";

  //OR

    String result = "";

    if(wgt<=0) result = "Error";
    if(wgt>0 && wgt < 10) result = "$3.00";
    if(wgt>=10 && wgt < 200) result = "$3.80";
    if(wgt>200) result = "Not a letter";

    return result;









}
```

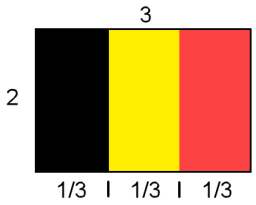
Question 3. Defining methods with parameters **[10 marks]**

The program on the facing page has two methods. The testDrawFlag() method calls the drawFlag(...) method three times to draw three different flags.

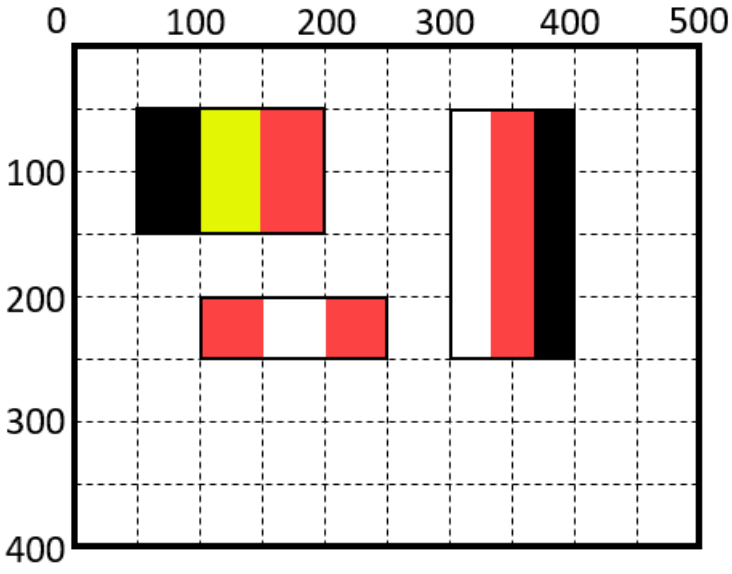
(a) **[7 marks]** Complete the drawFlag(...) method to draw a tri-color flag with a black rectangular border.

You will need to define seven parameters for drawFlag(...):

- the x, y position of the flag (i.e. top-left of the flag).
- the width of the flag.
- the ratio of the flag (height/width of the flag, e.g. 2.0/3.0 or 0.67 in the example below).
- the c1 (left), c2 (middle), and c3 (right) colors of the flag.



(b) **[3 marks]** Complete the testDrawFlag() method to draw three flags like the ones in the example picture below. The spotted lines should not be drawn. They are in the picture to indicate the positions and sizes of the flags. The flags align with the spotted lines. (The ratio of the flags does not need to be precisely the same as in the picture, e.g. there is no need to round results from division or multiplication).



The colors of the flags are (from left to right)

Flag	Colors
top/left	Color.black, Color.yellow, and Color.red
bottom/left	Color.red, Color.white, and Color.red
rightmost	Color.white, Color.red, and Color.black

```
public class Drawer {


    // Draws one flag at given position, width, ratio, and colors.
    public void drawFlag(   double x, double y, double width, double ratio,
                            Color c1, Color c2, Color c3
                                                                        ) {


        double height = width * ratio;
        double barWidth = width/3.0;

        UI.setColor(c1);
        UI. fillRect (x,y,barWidth,height);
        UI.setColor(c2);
        UI. fillRect (x+barWidth,y,barWidth,height);
        UI.setColor(c3);
        UI. fillRect (x+2*barWidth,y,barWidth,height);

        UI.setColor(Color.black);
        UI.drawRect(x,y,width,height);
    }
    // Draws three flags.
    public void testDrawFlag() {
        UI.clearGraphics ();
        this.drawFlag(  50,50,150,2.0/3.0, Color.black, Color.yellow, Color.red


                                                                        );
        this.drawFlag(  100,200,150,1.0/3.0, Color.red, Color.white, Color.red


                                                                        );
        this.drawFlag(  300,50,100,4.0/2.0, Color.white, Color.red, Color.black


                                                                        );

    }

}
```

Question 4. Writing methods that use objects **[10 marks]**

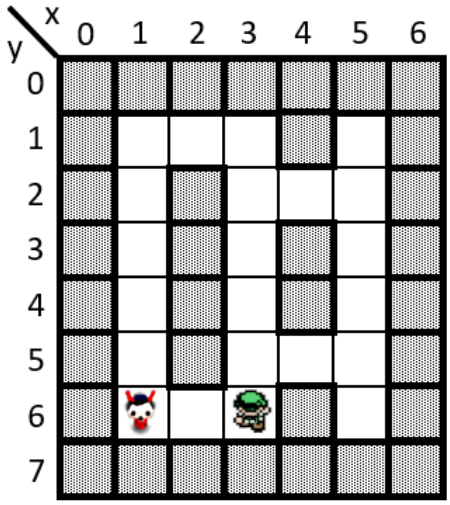
This question is about a program that creates and animates two ghosts in a simple 7 by 8 grid. The documentation for the Ghost class is given below.

```
class Ghost {
// Constructor to make a new Ghost object
    public Ghost(String name, int x, int y)
    /**The name parameter can only be either "HERO" or "EVIL".
       Creates a new Ghost object with a specified name at the (x,y) position in the grid.
       By default, the Ghost object faces upward in the grid */

// Methods that can be called on a Ghost object
    public void turn(String dir)
    /**The dir parameter can only be either "LEFT" or "RIGHT"
       Turns this object left or right (90 degrees) */

    public void move(int step)
    /** step > 0, the object moves step number of squares in the direction it faces.
        step =<0, the object stays in the current position
        If the ghost object moves into a solid block (like (0,0)) it "dies" and cannot move */

    public void attack(Ghost other)
    /** Attack the other ghost.
        A successful attack happens when the other ghost is at the same position */
}
```

Complete the following animate() method in the GhostGame class so that it

1. Creates two Ghosts objects in the game grid. A hero at position (3,6) and an evil at position (1,6). The game grid below is showing the game at this point.
2. Move the ghosts in the following order (and without any ghosts "dying")
   - Move the hero ghost to (3,1) using one method call
   - Moves the evil ghost to (3,5) using four method calls
   - Move the hero ghost to the evil ghost and perform a successful attack.

You **should** use methods in the Ghost class to do all the steps above, i.e. there is no need to use loops or conditionals to achieve the actions.



```
public class GhostGame{
    public void animate(){

            Ghost hero = new Ghost("HERO",3,6);
            Ghost evil  = new Ghost("EVIL",1,6);

            hero.move(5);

            evil.turn("RIGHT");
            evil.move(2);
            evil.turn("LEFT");
            evil.move(1);

            hero.turn("LEFT");//could be right as well
            hero.turn("LEFT");//must be right here, if right above
            hero.move(4);
            hero.attack( evil );

    }
}
```

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 5. Writing methods with for **[10 marks]**

The program for this question has three methods:

- analyseNumbers(), which asks the user for a sequence of negative numbers and then shows the result of calling the other two methods.
- findMinimum(...)
- plotNumbers(...)

(a) **[3 marks]** Complete the findMinimum(...) method to find and return the minimum number of the given ArrayList.
If the ArrayList is empty return 0

```
public void analyseNumbers () {
    ArrayList<Double> nums = UI.askNumbers("Enter numbers [-350, 0]");
    UI.println ("The minimum is " + this.findMinimum(nums));
    this.plotNumbers(nums);
}

public double findMinimum(ArrayList<Double> list) {


    double minimum = 0;

    for (double num: list ) {
            if (minimum>num) minimum = num;
    }

    return minimum;









}
```

(Question 5 continued on next page)

(Question 5 continued)

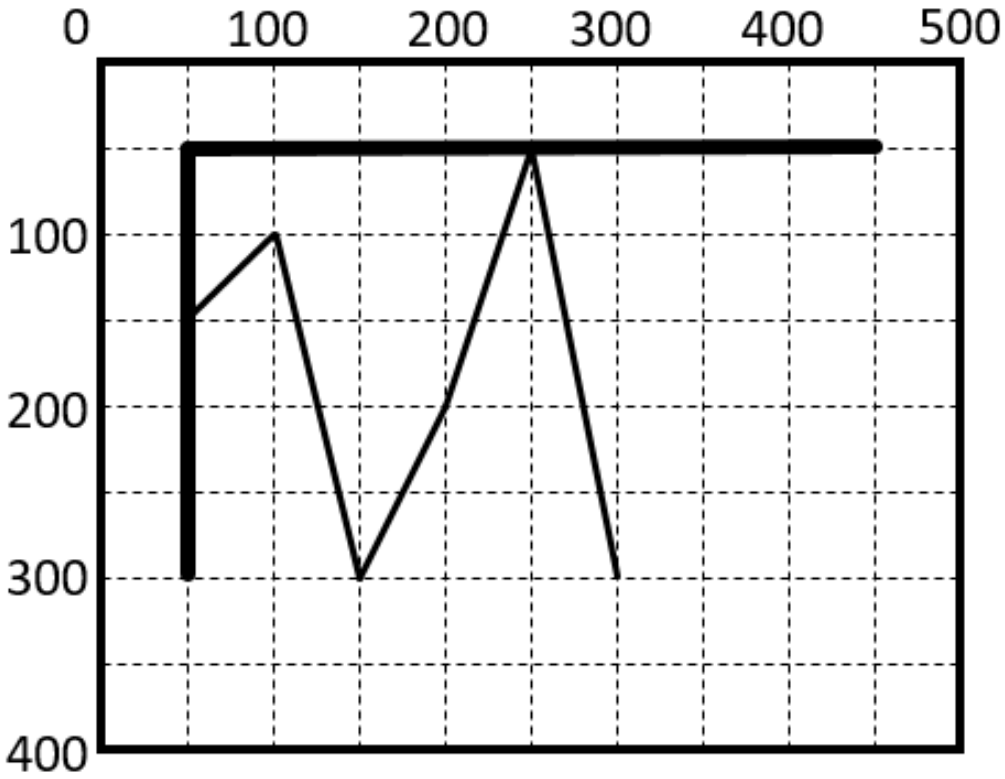
(b) **[7 marks]** Complete the plotNumbers(...) method to draw a line plot of the numbers in a graph.

- Draw the x-axis and the y-axis.
  - The origin should be at (50,50).
  - Negative values are below the x-axis.
  - The y-axis must end at the minimum. For instance, if the lowest number is -250, the y-axis stops at 300 on the graphic pane, which is the translation of -250.
- Draw the points every 50 units. If the first value is -100, then the first point on the graph is (0,-100), which translates to (50,150) in the graphics pane.
- You may assume the user enters at least one and at most nine numbers, all in the range of [-350, 0].
- If the user enters only one number, then the number will be shown by the length of the y-axis.
- The picture below uses different pen sizes for axis and the graph. You do not need to do this.
- The dotted lines are in the picture easing readability. You do not need to draw those.

For example, if the user enters the following numbers:

```
-100
-50
-250
-150
0
-250
done
```

There are two minimum positions at -250, and the plot should be like this:



(Question 5 continued on next page)

(Question 5 continued)

```
public void plotNumbers(ArrayList<Double> list){
    UI.clearGraphics();

    //Find minimum value in the list
    double minimum = findMinimum(list);

    //Draw axis
    UI.drawLine(50, 50, 450, 50);
    UI.drawLine(50, 50, 50, 50 - minimum);

    //Setup first point
    double prevX = 0; //Point is y-axis - 50 so first iteration aligns with y-axis
    double prevY = list.get(0);

    //loop all values
    for(double y : list)
    {
        //Get the previous point, if this is the first point, draw a dot
        double x = prevX + 50;
        if(x == 50) prevX = x;

        //Draw a line
        UI.drawLine(prevX, 50 - prevY, x, 50 - y);

        //Prepare for the next iteration
        prevX = x;
        prevY = y;
    }









}
```

* * * * * * * * * * * * * * *

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.