Isaac Ray Shoebottom
CS 1073 (FR02A)
Assignment 2
3429069

Section A

Output:

```
Did you get paid this week?
Yes
Did you buy groceries this week?
No
Do you have leftovers at home?
Yes
Are there enough leftovers for a meal?
No
You should eat at a restaurant

Process finished with exit code 0
```

```
Did you get paid this week?
Yes
Did you buy groceries this week?
Yes
You should eat at home.

Process finished with exit code 0
```

```
Did you get paid this week?
No
You should eat at home.

Process finished with exit code 0
```

Section B

Source Code (Main.java):

```
/**
 * @author Isaac Shoebottom (3429069)
 */

import java.util.Scanner;

public class Main {
```

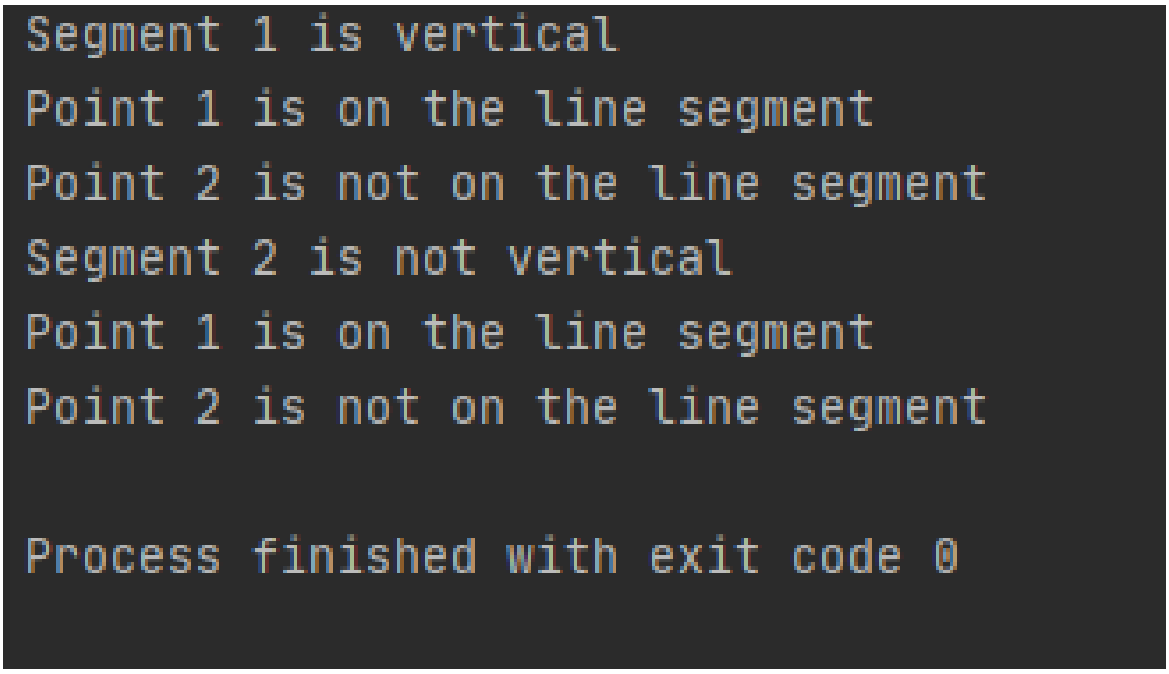
```
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        boolean gotPaid;
        boolean boughtGroceries;
        boolean leftoversAtHome;
        boolean enoughLeftoversForMeal;
        String scannerIn;

        System.out.println("Did you get paid this week?");
        scannerIn = scanner.nextLine();
        scannerIn = scannerIn.toLowerCase();
        scannerIn = (scannerIn.equals("yes")) ? "true" :
(scannerIn.equals("no")) ? "false" : "not yes or no";
        gotPaid = Boolean.parseBoolean(scannerIn);

        if (gotPaid) {

            System.out.println("Did you buy groceries this week?");
            scannerIn = scanner.nextLine();
            scannerIn = scannerIn.toLowerCase();
            scannerIn = (scannerIn.equals("yes")) ? "true" :
(scannerIn.equals("no")) ? "no" : "not yes or no";
            boughtGroceries = Boolean.parseBoolean(scannerIn);

            if (!boughtGroceries) {

                System.out.println("Do you have leftovers at home?");
                scannerIn = scanner.nextLine();
```

```
                scannerIn = scannerIn.toLowerCase();
                scannerIn = (scannerIn.equals("yes")) ? "true" :
(scannerIn.equals("no")) ? "no" : "not yes or no";
                leftoversAtHome = Boolean.parseBoolean(scannerIn);

                if (leftoversAtHome) {

                    System.out.println("Are there enough leftovers for
a meal?");
                    scannerIn = scanner.nextLine();
                    scannerIn = scannerIn.toLowerCase();
                    scannerIn = (scannerIn.equals("yes")) ? "true" :
(scannerIn.equals("no")) ? "no" : "not yes or no";
                    enoughLeftoversForMeal =
Boolean.parseBoolean(scannerIn);

                    if (!enoughLeftoversForMeal) {
                        System.out.println("You should eat at a
restaurant.");
                    }
                    else {
                        System.out.println("You should eat at home.");
                    }
                }
                else {
                    System.out.println("You should eat at a
restaurant.");
                }
            }
            else {
                System.out.println("You should eat at home.");
            }
```

```
        }
        else {
            System.out.println("You should eat at home.");
        }
    }
}
```

Section C

Output:



Section D

Source Code (TestLine.java):

```
/**
 * @author Isaac Shoebottom (3429069)
 */
public class TestLine {
```

```
public static void main(String[] args) {

    LineSegment segment1 = new LineSegment(1.0, 1.0, 5.0, 5.0);
    CartesianPoint point1s1 = new CartesianPoint(3.0, 3.0);
    CartesianPoint point2s1 = new CartesianPoint(2.0, 3.0);


    LineSegment segment2 = new LineSegment(2.0, 2.0, 2.0, 6.0);
    CartesianPoint point1s2 = new CartesianPoint(2.0, 4.0);
    CartesianPoint point2s2 = new CartesianPoint(1.0, 5.0);

    if (segment1.isVertical()) {
        System.out.println("Segment 1 is vertical");
    }
    else {
        System.out.println("Segment 1 is not vertical");
    }

    if (segment1.containsPoint(point1s1)){
        System.out.println("Point 1 is on the line segment");
    }
    else {
        System.out.println("Point 1 is not on the line segment");
    }

    if (segment1.containsPoint(point2s1)){
        System.out.println("Point 2 is on the line segment");
    }
    else {
        System.out.println("Point 2 is not on the line segment");
```

```
        }

        if (segment2.isVertical()) {
            System.out.println("Segment 2 is vertical");
        }
        else {
            System.out.println("Segment 2 is not vertical");
        }

        if (segment2.containsPoint(point1s2)){
            System.out.println("Point 1 is on the line segment");
        }
        else {
            System.out.println("Point 1 is not on the line segment");
        }

        if (segment2.containsPoint(point2s2)){
            System.out.println("Point 2 is on the line segment");
        }
        else {
            System.out.println("Point 2 is not on the line segment");
        }
    }
}
```

Source Code (LineSegment.java) :

```
/**
 This class represents a 2D line segment using 2 points.
 @author Natalie Webber
 @author Scott Bateman
```

```
 @author Isaac Shoebottom (3429069)
*/
public class LineSegment {

  private CartesianPoint pointA;
  private CartesianPoint pointB;



  public LineSegment (double x1, double y1, double x2, double y2) {
    pointA = new CartesianPoint (x1, y1);
    pointB = new CartesianPoint (x2, y2);
  }

  public LineSegment (CartesianPoint p1, CartesianPoint p2) {
    pointA = p1;
    pointB = p2;
  }

  public double getLength () {
    return pointA.distance(pointB);
  }

    /**
     * This method checks the cross product and dot products of the
line and the given point to check if the point p is on the segment
     * @param p The point that is being tested to be on the segment
     * @return Value of if the returned point is on the segment
     */
  public Boolean containsPoint (CartesianPoint p) {
      double crossProduct;
```

```
      crossProduct = ((p.getY() - pointA.getY()) * (pointB.getX() -
pointA.getX())) - ((p.getX() - pointA.getX()) * (pointB.getY() -
pointA.getY()));

      if (Math.abs(crossProduct) > Math.ulp(1.0)) {

        return false;

      }

      double dotProduct;

      dotProduct = ((p.getX() - pointA.getX()) * (pointB.getX() -
pointA.getX())) + ((p.getY() - pointA.getY()) * (pointB.getY() -
pointA.getY()));

      if (dotProduct < 0 ) {

        return false;

      }

      double squaredLength;

      squaredLength = ((pointB.getX() - pointA.getX()) *
(pointB.getX() - pointA.getX())) + ((pointB.getY() - pointA.getY()) *
(pointB.getY() - pointA.getY()));

      if (dotProduct > squaredLength) {

        return false;

      }

      return true;

  }


    /**

     * Method to check if the line is vertical (only on one point in
the x axis)

     * @return The value of is the line is vertical or not

     */

  public boolean isVertical() {

    if (pointA.getX() == pointB.getX()) {

      return false;

    }

    else {
```

```
        return true;
      }
    }
  }
```