CS1020 Mid-term Test
AY2012/2013 Semester 2

SECTION A


SECTION B

13. \[10, 9, 8, 7, 6, 5, 4, 3, 2, 1\]

14(a)

\[
\begin{array}{c}
\text{head} \\
10 \quad 20 \quad 8 \quad 15 \quad 7 \quad \ldots
\end{array}
\]

14(b) Idea: Traverse each list till the last node. If the references of the last nodes in both list are the same, this implies that the lists are “joined” at some place.

```java
boolean isJoined(ListNode<Integer> list1,
                 ListNode<Integer> list2) {
    ListNode<Integer> tail1 = list1;
    while (tail1.getNext() != null) {
        tail1 = tail1.getNext();
    }

    ListNode<Integer> tail2 = list2;
    while (tail2.getNext() != null) {
        tail2 = tail2.getNext();
    }
    return tail1 == tail2;
}
```
import java.awt.*;

class Rectangle {
   // Attributes: corner1: bottom-left; corner2: top-right
   Private Point corner1, corner2;

   // Constructors
   // Default constructor creates a rectangle at corners (0,0) and (1,1)
   // You are to write a single statement using ‘this’
   public Rectangle() { // 2 marks
      this(new Point(), new Point(1,1));
   }

   // Constructor to create rectangle at corners indicated by pt1 and pt2
   // You should call the setCorner() method
   public Rectangle(Point pt1, Point pt2) { // 2 marks
      setCorner(1, pt1);
      setCorner(2, pt2);
   }

   // Set respective corner (which==1: corner1; which==2: corner2)
   // You may assume that which is either 1 or 2
   public void setCorner(int which, Point pt) {
      if (which == 1)
         corner1 = pt;
      else
         corner2 = pt;
   }

   // Get respective corner (which==1: corner1; which==2: corner2)
   // You may assume that which is either 1 or 2
   public Point getCorner(int which) { // 4 marks
      return (which == 1) ? corner1 : corner2;
   }
}
```java
// Overriding toString() method
public String toString() {
    return "{{" + getCorner(1).x + "," + getCorner(1).y + ":{" +
        getCorner(2).x + "," + getCorner(2).y + 
    }}";
}

// Overriding equals() method
public boolean equals(Object obj) { // 6 marks
    if (obj instanceof Rectangle) {
        Rectangle rect = (Rectangle) obj;
        return this.getCorner(1).equals(rect.getCorner(1))
            &&
            this.getCorner(2).equals(rect.getCorner(2));
    } else
        return false;
}
```

Many students did not use the equals() method defined in the Point class. Instead they used the getX() and getY() methods to access the x- and y-coordinates of the corners (or they use .x and .y to access the public attributes x and y directly).
Complete the following TestRectangle.java program. You should not modify any code that is given, or add any method that is not shown in the program. Read the comments above each method to understand what is expected.

```java
import java.util.*;
import java.awt.*;

class TestRectangle {
    public static void main(String[] args) {
        ArrayList<Rectangle> rectangles = readInput();
        checkDuplicate(rectangles);
        System.out.println(rectangles);
    }

    // Read in data for a list of rectangles 8 marks
    public static ArrayList<Rectangle> readInput() {
        Scanner sc = new Scanner(System.in);
        Point pt1, pt2;

        ArrayList<Rectangle> list
            = new ArrayList<Rectangle>() {
        while (sc.hasNext()) {
            pt1 = new Point(sc.nextInt(), sc.nextInt());
            pt2 = new Point(sc.nextInt(), sc.nextInt());
            list.add(new Rectangle(pt1, pt2));
        }
        return list;
    }

    Common mistakes:
    Many students did not construct the 2 Point objects needed for the rectangle.
    Some students did not construct the Rectangle object as well.
```
public static void checkDuplicate(ArrayList<Rectangle> rectangles) {
  int size = rectangles.size();
  Rectangle lastRect = rectangles.get(size-1);

  // Check if last rectangle in list is a duplicate
  for (int i=0; i<size-1; i++) {
    if (lastRect.equals(rectangles.get(i))) {
      rectangles.remove(size-1);
      break;
    }
  }
}

Alternatively:

int size = rectangles.size();
Rectangle lastRect = rectangles.get(size-1);

// Check if last rectangle in list is a duplicate
if (rectangles.indexOf(lastRect) != size-1)
  rectangles.remove(size-1);

Other alternative answers are possible.