NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

Practical Examination for Semester 1, 2005/2006
CS1101X — Programming Methodology

22 October 2005 Time Allowed: 2 hours 30 minutes

INSTRUCTION TO CANDIDATES

1. This is an **OPEN book** practical examination. You are allowed to bring to the lab hard copies of notes and books. However, you are **NOT** allowed to bring into the lab digital/communication devices (such as laptop, cell phone, PDA, etc.) and storage devices (hard disk, thumb drive, CD, etc.). There are **2** tasks in **8** printed pages. The marking scheme for each task is as given – you should exercise the usual good programming practices (comments, proper indentation, meaningful naming convention, good design of classes etc.).

2. **Login to PC.** Please use the given ID (**cs1101xPE**) and password (**cs1101xPETHE900**) to log into your assigned PC with domain as **Computing**. Your NUSNET id will not be used as there is no connection to NUSNET. The PC comes with our recommended IDE **DrJava**. Java **manuals** are available on your desktop. We assume you have tried working with the assigned PC before the practical exam, and we will not provide software that are not already in the PC and we do not guarantee the availability of such software/editor. Please login to the CourseMarker before invoking DrJava.

3. **Login to CourseMarker.** Click on the CourseMarker shortcut **c:\cmc\cs1101xPE.bat** to log into the CourseMarker. Please use the **newly assigned password** (available to you just before the practical exam) to perform the login. Once you have gained access to the CourseMarker, please click on the setup button to download the necessary files and some dummy files – please **do not** delete any of these files though you are not using them in your programming, you need them when submitting your work to the CourseMarker. With this, you are ready to start programming.

4. **Working Directory.** Please create your files in the directory **"c:\Documents and Settings\cs1101xPE\My Documents\CMhome\studentArea\yourID\cs1101xPE"** (where **yourID** is your CourseMarker login id) with the specified file names as stated in the questions. Only the **specified files** in the **specified directory** will be submitted to the CourseMarker when you click on the submit button of the CourseMarker client.

5. **Submission.** You can submit your work as often as you like. Only the **last version** will be graded. Note once again that the system **only** submits files of the specified filenames in the specified directory – all other files will not be captured by the CourseMarker. Please ensure you have submitted your work before the end of the examination – to avoid unnecessary stress to yourself, you should not wait till the very last minute of the exam to submit the work at the same time as many other students. The CourseMarker system will be closed shortly after the practical examination.

6. **Leaving the lab.** Please **log out** from the PC before you leave the lab. But please **do not** shut down the machine.

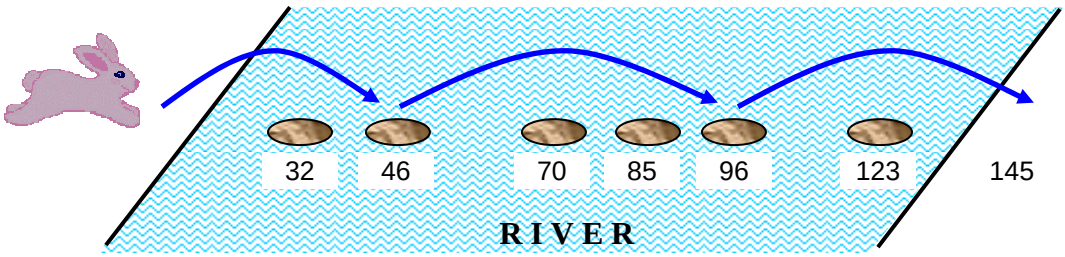
7. **No Communication.** You are NOT to communicate in any way (sharing of files, PCs, calculators, send emails, receive emails, use of ICQ, etc.) with anyone, other than the invigilators, during the practical examination. You must allow the invigilator access to your PC on request.

ALL THE BEST!

CS1101X AY2005/6 Semester 1
Practical Exam
(Date: 22 October 2005, Saturday)

Task 1: Rabbit Jumps (RabbitJumps.java): 40%

A bunny can hop at most 50 centimetres far. It wants to cross to the other side of the river, but it cannot swim. So the only hope is to hop on the rocks on the river, which are positioned in a straight line. The positions of the rocks are measured from the start location, assuming that the bunny starts at location zero. The opposite bank could be treated as a big rock. In the diagram below, the rocks are at locations 32, 46, 70, 85, 96, 123, and the opposite riverbank at location 145.



The bunny will jump as far as it could for each hop. **What is the smallest number of jumps it needs to take to reach the other side of the river?** For the above example, it needs to make 3 jumps, as shown in the diagram above.

Write a program **RabbitJumps.java** that reads in a positive integer ***n*** that represents the number of rocks (including the opposite riverbank), and then on the next line *n* distinct positive integers in increasing order that represent the locations of the rocks. Your program should output the minimum number of jumps needed, or –1 if it is not possible for the bunny to reach the other side of the river.

There will be at most 20 rocks (including the opposite bank).

Some sample runs are shown below.

Sample run #1:

```
$ javac RabbitJumps.java
$ java RabbitJumps
Enter n: 7
32 46 70 85 96 123 145
Answer: 3
```

Sample run #2:

```
$ java RabbitJumps
Enter n: 5
40 70 150 160 180
Answer: -1
```

Sample run #3:

```
$ java RabbitJumps
Enter n: 11
30 70 75 120 160 170 180 190 200 246 258
Answer: 7
```

Important notes

- You are to use an integer array to store the rock positions.
- You are to submit a program that can be compiled. You will lose 25 marks out of 50 marks right away if your program cannot be compiled (5 marks for compilability; 20 marks for correctness).
- Do not spend more than 1 hour on this task.
- You are advised to spend some time thinking over the problem to design your algorithm, instead of writing the code right away.
- A template program **RabbitJumps.java** is provided on the next page.

Marking scheme: Total of 50 marks

1. Can program be compiled? 5 marks
2. Use of array and values read correctly into array? 10 marks
3. Style (including your particulars, use of constant, comments, spacing/indentation, choice of variable names): 15 marks
4. Correctness of result: 20 marks
    a. 10 test data sets.
    b. 2 marks for each data set.
5. The marks for this task will constitute 40% of the total marks.

```java
// CS1101 (AY2005/6 Semester 1)
// PE Task 1
// RabbitJumps.java
// Author:
// Matriculation number:
// Discussion group:
/**
 * A bunny can jump at most 50 centimetres. It needs to cross a
 * river, with rocks positioned in a straight line in the river.
 * This program computes the minimum number of jumps for the
 * rabbit to cross to the other side of the river.
 */

import java.util.*;

public class RabbitJumps {


   public static void main (String[] args) {

      // fill in your code here

   }
}
```

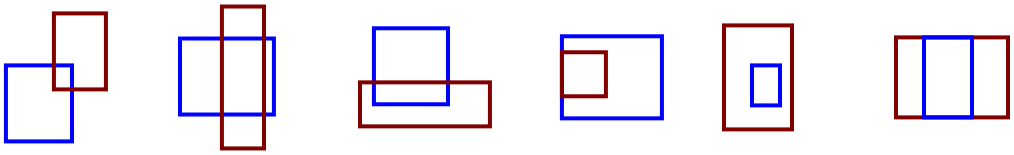
Task 2: Rectangles (MyRectangle.java): 60%

Two rectangles *A* and *B*, whose sides are parallel to the x-axis and y-axis, can be found to be in one of the following 3 cases:

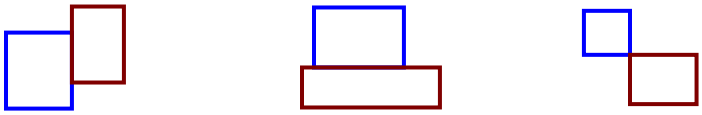
- Case 1: Rectangles *A* and *B* overlap each other.
- Case 2: Rectangles *A* and *B* touch each other.
- Case 3: Rectangles *A* and *B* are disjoint.

The diagrams below show the various positions of the rectangles in the 3 cases.

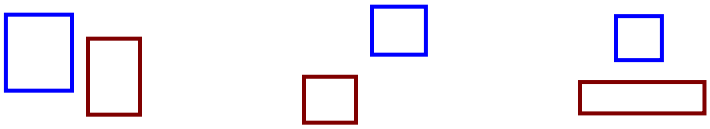
Case 1: Rectangles *A* and *B* overlap each other.



Case 2: Rectangles *A* and *B* touch each other.



Case 3: Rectangles *A* and *B* are disjoint.



Write a program **MyRectangle.java** to create the **MyRectangle** class that includes two data members: the bottom-left corner and the top-right corner of the rectangle, both of which are objects of the **Point2D.Double** class. You are also to design relevant methods for the following requirements.

The main method in your program should create 2 MyRectangle objects from user's inputs, print the rectangles' information, and output a conclusion that states whether the two rectangles **overlap**, **touch**, or are **disjoint**. Please refer to the 3 sample runs below.

If the two rectangles overlap each other, you need also to compute and display the overlapped area. Please refer to the first sample run below for an example.

You are to note that in the creation of a rectangle, the user may enter any 2 opposite corners of the rectangle, not necessarily the bottom-left corner and top-right corner. Therefore you need to check the values and make the necessary adjustments before creating the rectangle.

You may assume that the user enters two distinct points for the corners, and that the rectangle has a positive area.

You may write any additional methods you deem necessary. A more modular program will receive more credit than a less modular one.

Sample run #1:

```
$ javac MyRectangle.java
$ java MyRectangle
Enter one corner of 1st rectangle: 7.6 2.2
Enter opp corner of 1st rectangle: 1.5 0.3
Enter one corner of 2nd rectangle: 9.4 -2.5
Enter opp corner of 2nd rectangle: 4 4.2
Rectangle A = ([1.5, 0.3], [7.6, 2.2])
Rectangle B = ([4.0, -2.5], [9.4, 4.2])
Rectangles A and B overlap each other.
Overlapped area = 6.84
```

Sample run #2:

```
$ java MyRectangle
Enter one corner of 1st rectangle: 33.3 –1.1
Enter opp corner of 1st rectangle: 20.5 3.6
Enter one corner of 2nd rectangle: 20.5 8.6
Enter opp corner of 2nd rectangle: 10.3 –5.2
Rectangle A = ([20.5, -1.1], [33.3, 3.6])
Rectangle B = ([10.3, -5.2], [20.5, 8.6])
Rectangles A and B touch each other.
```

Sample run #3:

```
$ java MyRectangle
Enter one corner of 1st rectangle: 4 0
Enter opp corner of 1st rectangle: 9 5
Enter one corner of 2nd rectangle: 0 11
Enter opp corner of 2nd rectangle: 5 6
Rectangle A = ([4.0, 0.0], [9.0, 5.0])
Rectangle B = ([0.0, 6.0], [5.0, 11.0])
Rectangles A and B are disjoint.
```

Important notes

- You are to submit a program that can be compiled. You will lose 30 marks for correctness right away if your program cannot be compiled.
- Do not spend more than 1 hour 30 minutes on this task.
- You are advised to spend some time thinking over the problem to design your algorithm, instead of writing the code right away.
- Take note of the output format. A rectangle is displayed in this format:

  `([`*x1*`, `*y1*`], [`*x2*`, `*y2*`])`

  where *x1* and *y1* are the x- and y-coordinates of the bottom-left corner of the rectangle, and *x2* and *y2* are the x- and y-coordinates of the top-right corner.
- There is no need to use DecimalFormat to format the numerical values.
- A template program **MyRectangle.java** is provided on page 8.

Marking scheme: Total of 50 marks

1. Class definition (constructor, mutators, accessors, toString): 10 marks
2. Style (your particulars, comments, spacing/indentation, choice of variable names): 10 marks
3. Correctness of result: 30 marks
    a. 10 test data sets.
    b. 2 to 4 marks for each data set.
4. The marks for this task will constitute 60% of the total marks.

```
// CS1101 (AY2005/6 Semester 1)
// PE Task 2
// MyRectangle.java
// Author:
// Matriculation number:
// Discussion group:

import java.awt.geom.*;
import java.util.*;

public class MyRectangle {

   // Data members: the two corner points
   private Point2D.Double bottomLeftPoint;
   private Point2D.Double topRightPoint;

   // Constructor
   public MyRectangle(Point2D.Double pt1, Point2D.Double pt2) {
      // fill in the code here
   }

//-------------------------------------------------------------
// Methods:
//  (Each method should be preceded by a line of description.)
//  (You may create additional methods of your own.)
//-------------------------------------------------------------
   public void setBottomLeftPoint (???) {
      // fill in the code here
   }

   public void setTopRightPoint (???) {
      // fill in the code here
   }

   public ??? getBottomLeftPoint () {
      // fill in the code here
   }

   public ??? getTopRightPoint () {
      // fill in the code here
   }

   public String toString() {
      // fill in the code here
   }

   //-----------------------------------------------------
   // main method used to test the MyRectangle class
   //-----------------------------------------------------
   public static void main (String [] args)  {

      Scanner scanner = new Scanner(System.in);

      // fill in the code here
   }
}
```