

23rd Annual
Rowan University
Programming Contest

hosted by the
Computer Science Department

Friday, 3 April 2009

Contest Problem



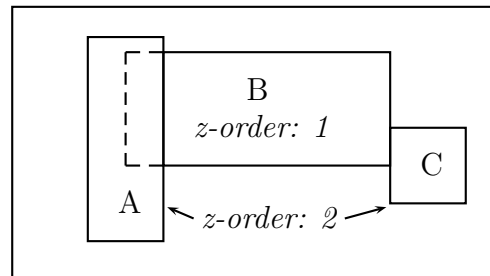
1 Background

Since the first development of window-based graphical user interfaces in the 1960s, any such system has had to deal with the problem of overlapping windows.

The windows on a computer desktop have a height and width, and the upper-left corner has coordinates which indicate where the window is on the screen. The two-dimensional screen is usually seen with the familiar **X** and **Y** axes of the Cartesian plane (except that the location (0,0) is the upper-left, not lower-left, corner). The windows are also given what's known as a **Z-order**, indicating which window is 'on top' of the others.

In Z-order, a higher number means that the window is higher up in the stack. A window with Z-order of 4 is above a window with Z-order 3. We will consider '0' to be the Z-order of the desktop icons, and all windows will have a Z-order which is a positive integer.

Window B has a z-order of 1.
 Windows A and C have a z-order of 2.
 Note that two windows may have the same z-order, provided they do not overlap.
 (This corresponds to the first data set in Sample Input #1.)



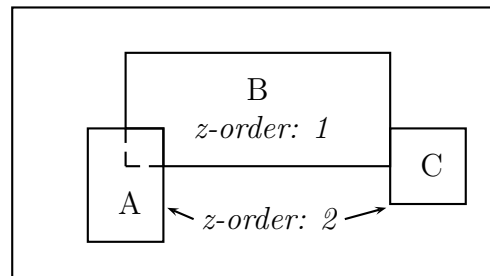
When windows overlap, there is the problem of what to do when a window on the 'bottom' of the stack is redrawn by its program. Suppose, in the diagram above, the program controlling Window B repaints it. If the *entire* rectangle is repainted, then Window A will be partially obscured, and will have to repaint as well to maintain the appearance that A is on top of B. This needless multiple repainting would inevitably be annoying to the end-user.

An alternative solution would be to maintain in memory both (a) the total client rectangle for a window, and (b) the *visible portion*, taking into account the Z-order of all the windows currently open, and updating this information when windows are moved, or a new window opens, or another is brought to the top.

Your challenge is to write a program that reads in window placement information, and prints out, for each window, the corners of that section which is actually visible to the user.

As a simplifying assumption, you may assume that any window in front of any other window will leave exposed a rectangular portion, or will cover the other window entirely. That is, you will **not** have to handle a case such as:

This situation is like the one above, but window B's area which is left exposed by window A is not rectangular. Obviously such situations can and do occur in actual window systems, but your program does not have to handle a situation of this kind.



You are also guaranteed that, for all data sets, the visible area of any window will be contiguous.

2 Input

2.1 Input Specification

For text input, your program should accept input in the following format:

1. An integer, \mathcal{D} , where $1 \leq \mathcal{D} \leq 50$, which is the number of datasets in this file.
2. \mathcal{D} data sets, each of which is in this format:
 - (a) Two integers, giving the width and height of the desktop.
 - (b) One integer, \mathcal{W} , where $1 \leq \mathcal{W} \leq 50$, giving the number of windows in the dataset.
 - (c) \mathcal{W} lines, each of which will have 5 integers, describing the window's placement.

The first two integers represent the X and Y coordinates of the upper-left corner.

The next two integers represent the width and height of the window.

The fifth integer represents the window's Z-order.

NOTES:

The width and height of the desktop, and the width and height of all windows, will be in the interval $[0,10000]$.

Z-orders will be in the interval $[1,1000]$.

2.2 Sample Input #1

Data in file	Item # above:	Meaning in plain English
2	1	<i>this file has 2 data sets</i>
1280 720	2a	<i>screen 1 is 1280×720 pixels</i>
3	2b	<i>data set 1 has 3 windows</i>
200 80 200 540 2	} 2c	<i>placement information for the three windows</i>
300 120 700 300 1		
1000 320 200 200 2		
320 480	2a	<i>screen 2 is 320×480</i>
4	2b	<i>data set 2 has 4 windows</i>
0 0 160 240 1	} 2b	<i>placement information for the four windows</i>
160 0 160 240 1		
0 240 160 240 1		
160 240 160 240 1		

(This input will be available on the website as **sample1.txt**.)

NOTES:

You may choose to have your program read the input from the keyboard, or ask the user for a filename and then read the file. Users of GUI-based programming environments may prefer to use text boxes into which the values can be entered, and buttons to begin their calculation. Any reasonable variation in the spirit of the problem is acceptable.

3 Output

3.1 Output Specification

For each data set configuration, your program must generate output as follows:

1. The text ‘Analyzing **D** data sets’, where **D** is the number of data sets in the input.
2. The text ‘Data set **S**:’, where **S** is the number of the data set being reported on.
3. The text ‘Screen size: **X** x **Y**’, where **X** is the width of the screen and **Y** is the height of the screen.
4. The text ‘Windows: **W**’, where **W** is the number of windows in the data set being reported on.
5. For each window, either the text ‘Window **N**: (**ULX**, **ULY**) to (**LRX**, **LRY**)’, where:
 - (a) **N** is the window’s number (*i.e.*, where it was on the input)
 - (b) **ULX** is the X coordinate of the upper-left corner of the visible part of the window
 - (c) **ULY** is the Y coordinate of the upper-left corner of the visible part of the window
 - (d) **LRX** is the X coordinate of the lower-right corner of the visible part of the window
 - (e) **LRY** is the Y coordinate of the lower-right corner of the visible part of the window

or the text

‘Window **N**: completely obscured’, if a window is not visible at all.

The windows should be printed out in the order they were read in. (So you’ll list ‘Window 1’, ‘Window 2’, ‘Window 3’, &c.)

3.2 Sample Output #1

Output Generated by Program	Item # above:	Meaning in plain English
Analyzing 2 Data sets	1	<i>there are two data sets in the file</i>
Data set 1:	2	<i>this is for data set 1</i>
Screen size: 1280 x 720	3	<i>the screen size</i>
Windows: 3	4	<i>data set 1 has three windows</i>
Window 1: (200, 80) to (399, 619)	5	<i>the visible area of window 1</i>
Window 2: (400, 120) to (999, 419)	5	<i>the visible area of window 2</i>
Window 3: (1000, 320) to (1199, 519)	5	<i>the visible area of window 3</i>
Data set 2:	2	<i>this is for data set 2</i>
Screen size: 320 x 480	3	<i>the screen size</i>
Windows: 4	4	<i>data set 2 has four windows</i>
Window 1: (0, 0) to (159, 239)	5	<i>the visible area of window 1</i>
Window 2: (160, 0) to (319, 239)	5	<i>the visible area of window 2</i>
Window 3: (0, 240) to (159, 479)	5	<i>the visible area of window 3</i>
Window 4: (160, 240) to (319, 479)	5	<i>the visible area of window 4</i>

(This output corresponds to the sample input from §2.2.)

4 Sample Data

4.1 Sample Input #2

Data in file	Item #	Meaning in plain English
4	1	<i>this file has 4 data sets</i>
1280 720	2a	<i>screen 1 is 1280×720 pixels</i>
2	2b	<i>data set 1 has 2 windows</i>
-100 100 200 100 1	2c	<i>placement information for the two windows</i>
1200 700 100 100 1		
800 600	2a	<i>screen 2 is 800×600</i>
5	2b	<i>data set 2 has 5 windows</i>
50 50 100 100 1	2b	<i>placement information for the five windows</i>
50 200 100 100 2		
200 50 100 100 3		
200 200 100 100 4		
0 0 800 600 5		
432 240	2a	<i>screen 3 is 432×240</i>
3	2b	<i>data set 3 has 3 windows</i>
50 50 100 100 1	2b	<i>placement information for the three windows</i>
100 50 100 100 2		
150 50 100 100 3		
1920 1080	2a	<i>screen 4 is 1920×1080</i>
3	2b	<i>data set 4 has 3 windows</i>
100 100 1000 500 5	2b	<i>placement information for the three windows</i>
300 200 200 200 2		
200 400 400 150 3		

4.2 Sample Output #2

Analyzing 4 data sets

Data set 1:

Screen size: 1280 x 720

Windows: 2

Window 1: (0,100) to (99,199)

Window 2: (1200,700) to (1279,719)

Data set 2:

Screen size: 800 x 600

Windows: 5

Window 1: completely obscured

Window 2: completely obscured

Window 3: completely obscured

Window 4: completely obscured

Window 5: (0,0) to (799,599)

Data set 3:

Screen size: 432 x 240

Windows: 3

Window 1: (50,50) to (99,149)

Window 2: (100,50) to (149,149)

Window 3: (150,50) to (249,149)

Data set 4:

Screen size: 1920 x 1080

Windows: 3

Window 1: (100,100) to (1099,599)

Window 2: completely obscured

Window 3: completely obscured

Notes on Sample Data #2:

All Data Sets: The screen's coordinates start at (0,0) in the upper-left corner, and that must be accounted for. An 800-pixel-wide screen only has X-coordinates from 0 to 799.

Data Set 1, Window 1: the X coordinate of the upper-left corner is negative, indicating that the window is partly off-screen to the left. (Note that window 2 in this data set is also partly off-screen, in the lower-right corner.) A window could theoretically be completely off-screen.

Data Set 3: The z-order for a window does not necessarily have anything to do with the order in which it is listed.

5 Test Data

Run your program on this input and print the results. **You must submit printed output to earn full points.** Your program will also be run on data known only to the judges.

5.1 Test Input #1

```

5
100 100
5
40 40 21 21 7
20 41 21 19 6
60 41 21 19 6
41 20 19 21 5
41 60 19 21 4
100 100
2
20 20 21 21 7
60 60 21 21 1
100 100
10
40 30 26 31 2
15 70 71 11 8
50 30 26 31 3
30 55 46 6 4
65 20 16 51 5
30 30 26 31 1
20 20 16 56 6
65 10 16 11 9
25 25 56 16 4
15 80 35 30 7
1920 1080
6
0 -20 50 10 573
-20 0 10 50 12
1920 10 10 10 4
10 1080 10 10 123
-1 100 2 10 7
50 -10 10 10 9
100 100
11
30 55 46 6 4
30 30 26 31 1
65 10 16 11 9
50 30 26 31 3
25 25 56 16 4
65 20 16 51 5
40 30 26 31 2
20 20 16 56 6
15 70 71 11 8
15 80 35 30 7
60 5 30 90 11

```

5.2 Test Input #2

```

1
200 200
50
-10 0 11 1 3
0 -10 1 11 4
199 0 5 1 5
100 -4 1 5 4
-1 199 2 1 9
0 199 1 2 10
199 199 3 1 2
199 199 1 5 1
-5 10 11 20 202
0 5 11 30 203
-5 45 11 15 193
0 40 6 25 201
20 0 20 11 504
25 -5 10 11 12
60 0 10 6 908
60 -5 10 6 98
80 -15 10 10 400
80 -15 10 10 100
80 -15 10 10 300
80 -15 10 10 200
80 -15 10 10 500
80 15 10 10 100
80 15 10 10 500
80 15 10 10 200
80 15 10 10 400
80 15 10 10 300
198 70 15 10 9
190 70 15 10 1
195 70 15 10 6
191 70 15 10 2
193 70 15 10 4
194 70 15 10 5
201 70 15 10 31
196 70 15 10 7
192 70 15 10 3
197 70 15 10 8
200 70 15 10 11
199 70 15 10 10
50 195 10 10 10
50 201 10 10 16
50 196 10 10 11
50 199 10 10 14
50 197 10 10 12
50 200 10 10 15
50 198 10 10 13
185 199 1 1 50
185 199 1 1 5
185 200 1 1 9
185 200 1 1 19
100 100 10 10 5

```

All sample and test data sets are available at
<<http://elvis.rowan.edu/rupc/2009>>.