# 2005 Canadian Computing Competition
# Day 2, Question 1

Input file: `primed.in`
Output file: `primed.out`
Source file: `n:\primed\primed.___`

## Primed Sequences

Given a sequence of positive integers of length $n$, we define a *primed subsequence* as a consecutive subsequence of length at least two that sums to a prime number greater than or equal to two.

For example, given the sequence:

```
3 5 6 3 8
```

There are two primed subsequences of length 2 ($5 + 6 = 11$ and $3 + 8 = 11$), one primed subsequence of length 3 ($6 + 3 + 8 = 17$), and one primed subsequence of length 4 ($3 + 5 + 6 + 3 = 17$).

**Input**

Input consists of a series of test cases. The first line consists of an integer $t$ ($1 \leq t \leq 20$), the number of test cases.

Each test case consists of one line. The line begins with the integer $n$, $0 < n \leq 10000$, followed by $n$ non-negative numbers less than 10000 comprising the sequence. You should note that 80% of the test cases will have at most 1000 numbers in the sequence.

**Output**

For each sequence, print the "Shortest primed subsequence is length $x$:", where $x$ is the length of the shortest primed subsequence, followed by the shortest primed subsequence, separated by spaces. If there are multiple such sequences, print the one that occurs first.

If there are no such sequences, print "This sequence is anti-primed.".

**Sample Input**

```
3
5 3 5 6 3 8
5 6 4 5 4 12
21 15 17 16 32 28 22 26 30 34 29 31 20 24 18 33 35 25 27 23 19 21
```

**Sample Output**

```
Shortest primed subsequence is length 2: 5 6
Shortest primed subsequence is length 3: 4 5 4
This sequence is anti-primed.
```

# 2005 Canadian Computing Competition
## Day 2, Question 2

Input file: `segments.in`
Output file: `segments.out`
Source file: `n:\segments\segments.___`

## Segments

You are to find the length of the shortest path from the top to the bottom of a grid covering specified points along the way.

More precisely, you are given an $n$ by $n$ grid, rows $1..n$ and columns $1..n$ ($1 \leq n \leq 20000$). On each row $i$, two points $L(i)$ and $R(i)$ are given where $1 \leq L(i) \leq R(i) \leq n$. You are to find the shortest distance from position $(1, 1)$, to $(n, n)$ that visits all of the given segments in order. In particular, for each row $i$, all the points

$$(i, L(i)), (i, L(i) + 1), (i, L(i) + 2), ..., (i, R(i)),$$

must be visited. Notice that one step is taken when dropping down between consecutive rows. Note that you can only move left, right and down (you cannot move up a level). On finishing the segment on row $n$, you are to go to position $(n, n)$, if not already there. The total distance covered is then reported.

### Input

The first line of input consists of an integer $n$, the number of rows/columns on the grid. On each of the next $n$ lines, there are two integers $L(i)$ followed by $R(i)$ (where $1 \leq L(i) \leq R(i) \leq n$).

### Output

The output is one integer, which is the length of the (shortest) path from $(1, 1)$ to $(n, n)$ which covers all intervals $L(i), R(i)$.

### Sample Input

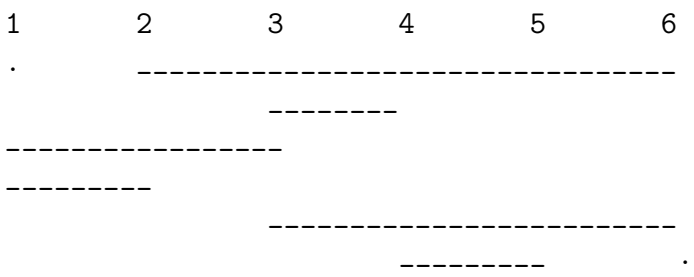
```
6
2 6
3 4
1 3
1 2
3 6
4 5
```

### Sample Output

```
24
```

**Explanation of Sample Input/Output**

Below is a pictoral representation of the input.




Notice that on the first row, we must traverse 5 units to the right and then drop down one level.

On the second row, we must traverse 3 units to the left and drop down one level.

On the third row, we must traverse 2 units to the left and drop down one level.

On the fourth row, we move 1 unit to the right and then drop down one level.

On the fifth row, we move 4 units to the right and drop down one level.

On the sixth (and final) row, we move 2 units left, then 2 units right.

In total, we have moved $6 + 4 + 3 + 2 + 5 + 4 = 24$ units.

# 2005 Canadian Computing Competition
# Day 2, Question 3

Input file: `windows.in`
Output file: `windows.out`
Source file: `n:\windows\windows.___`

## Move that Mouse AGAIN

You probably remember from Stage 1 that computers have a *mouse* and sometimes, if you really have to, you have a bunch of *windows* that you can click on.

In particular, you will have a screen which has $R$ ($1 \leq R \leq 10000$) rows and $C$ ($1 \leq C \leq 10000$) columns. You have $n$ ($1 \leq n \leq 50000$) rectangular windows, defined by the "top-left" coordinates $(x_l, y_t)$ and "bottom-right" coordinates $(x_r, y_b)$. Note that you can assume that $1 \leq x_l < x_r \leq C$ and $1 \leq y_b < y_t \leq R$ (thus, you don't have empty windows). It is worth noting that a window includes all points on its borders: that is, the window defined by $(x_l, y_t)$ and $(x_r, y_b)$ includes exactly the points $(x, y)$ such that $x_l \leq x \leq x_r$ and $y_b \leq y \leq y_t$.

If two windows overlap, the one which is on top of the other window will be listed *later* in the sequence (not necessarily immediately after, however).

Here comes the mouse part.

You also have a mouse that can click on windows. Initially, the mouse is at position $(1, 1)$ (the bottom-left corner of the screen). The mouse will then to be told to move to a particular position (say $(x, y)$, where $1 \leq x \leq C$ and $1 \leq y \leq R$) and click. When the mouse clicks, the window which is visible at that position moves to the top (that is, the entire window becomes visible, possibly hiding some other windows). There will be $m$ ($1 \leq m \leq 10000$) such mouse clicks.

Your job is to indicate which of the $n$ windows is "in focus" (i.e., most recently clicked on) after each mouse move.

**Input**

The first line of input consists of an integer $C$, the number of columns. The second line of input consists of an integer $R$, the number of rows. The third line of input consists of the number $n$, the number of windows. On each of the next $n$ lines, there are four integers $x_l$ followed by $y_t$ followed by $x_r$ followed by $y_b$, indicating the "top-left" and "bottom-right" coordinates of this window. (These windows are numbered from 1 to $n$). On the next line is the integer $m$. On each of the next $m$ lines will be two integers, $x$ followed by $y$, indicating the new position of the mouse.

**Output**

The output is $m$ lines long, each line containing an integer $v_i$ such that $0 \leq v_i \leq n$. If line $i$ contains the integer $v_i$ ($v_i > 0$), that indicates that after the $i$th mouse move, the window

$v_i$ was just clicked on (and has moved to the top). If, instead, $v_i = 0$, then there was no window at that position.

### Sample Input

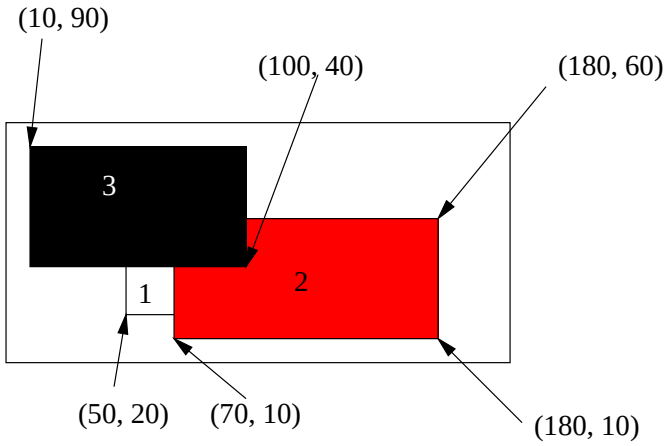
```
200
100
3
50 50 80 20
70 60 180 10
10 90 100 40
3
60 20
150 90
150 30
```

### Sample Output

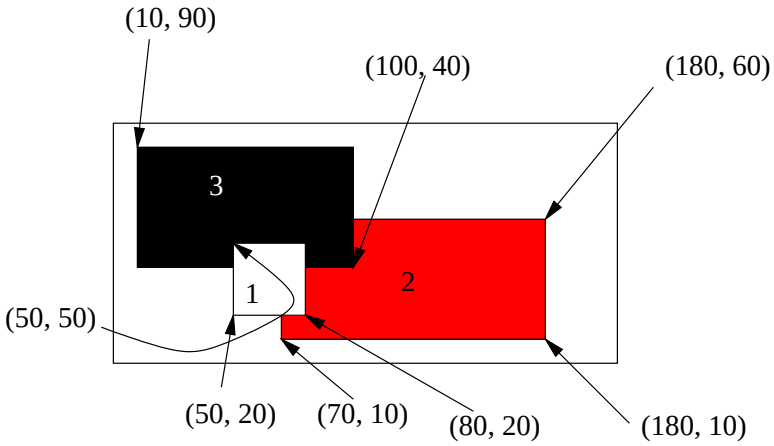
```
1
0
2
```

### Explanation of Sample Input/Output

The screen has width 200 and height 100. There are three windows on the screen, and initially, we have the following configuration:




After the mouse clicks at $(60, 20)$, we have




Since there is no window at position $(150, 90)$, 0 is outputted.

Once we click at position $(150, 30)$, that moves window 2 into focus, to get the following picture.