South African Computer Olympiad
Final round
Day 2

Overview

Author	Harry Wiggins and Carl Hultquist (Originally from USACO 1999 National Championship)	Carl Hultquist	Bruce Merry
Problem	**barn**	**break**	**supplies**
Source	barn.pas barn.c barn.cpp barn.java	N/A	supplies.pas supplies.c supplies.cpp supplies.java
Input file	barn.in	break.in	stdin
Output file	barn.out	break.out	stdout
Time limit	2 seconds	N/A	2 seconds
Number of tests	10	10	10
Points per test	10	10	10
Total points	**100**	**100**	**100**

The maximum total score is 300 points.

Biggest Barn

Author

Harry Wiggins and Carl Hultquist (Originally from USACO 1999 National Championship)

Introduction

Farmer John wants to place a big square barn on his square farm. He hates to cut down trees on his farm and wants to find a location for his barn that enables him to build it only on land that is already clear of trees. For our purposes, his land is divided into $N \times N$ parcels. The input contains a list of parcels that contain trees.

Task

Your job is to determine and report the largest possible square barn that can be placed on his land without having to clear away trees. The barn sides must be parallel to the horizontal or vertical axis.

Example

Consider the following grid of Farmer John's land where '.' represents a parcel with no trees and '#' represents a parcel with trees:

```
  1 2 3 4 5 6 7 8
1 . . . . . . . .
2 . # . . . # . .
3 . . . . . . . .
4 . . . . . . . .
5 . . . . . . . .
6 . . # . . . . .
7 . . . . . . . .
8 . . . . . . . .
```

The largest barn is 5×5 and can be placed in either of two locations in the lower right part of the grid.

Input (barn.in)

The first line of input will contain two integers, N and T. N is the number of parcels per side of the farm, and T is the number of parcels with trees. The next T lines will each contain two integers, R and C, which describe the row and column respectively of each tree parcel.

Sample input

```
8 3
2 2
2 6
6 3
```

Output (barn.out)

The output file should consist of exactly one line, the maximum side length of John's barn.

Sample output

```
5
```

Constraints

- $1 \leq N \leq 1000$
- $1 \leq T \leq 10000$
- $1 \leq$ each row and column of every tree $\leq N$

50% constraints

- $1 \leq N \leq 50$
- $1 \leq T \leq x300$
- $1 \leq$ each row and column of every tree $\leq N$

Time limit

2 seconds.

Scoring

If your program's output for a test-case does not conform to the output format described above, then you will score 0 for that test-case. Otherwise, you will score 10 points if your program outputs the correct answer and 0 points if it outputs the incorrect answer.

Prison Break

Author

Carl Hultquist

Introduction

Many years ago, Bruce was falsely convicted for creating a computer worm that would syphon money from other people's bank accounts into his own. Since then, he's been locked up in prison, waiting for the right moment when he can make his escape and clear his name.

That moment has arrived: the guards are all distracted, and Bruce has managed to pick the lock to his cell! Now he just needs to get out of the prison...

To do this, Bruce will need to visit as many rooms in the cell-block as he can looking for the master key which will let him out the main door. The more rooms he can visit, the better his chances of finding the key. Since Bruce has been in the prison for a long time, he knows exactly how many rooms there are and how they're connected to each-other by doorways.

The rooms are also wired with motion sensors, which will go off as soon as Bruce walks through them: when they do, the guards will be alerted and will start running towards the room that set off the alarm. For this reason, Bruce **cannot** visit any room more than once, for if he did so he would likely run into the guards and be re-imprisoned!

Task

Your task is to find a route that Bruce should take so that he visits as many rooms in the cell-block as possible.

Example

Suppose the cell-block has the layout shown in figure 1.

The `B` indicates where Bruce is currently, and the numbers indicate doorways between rooms.

In this example, Bruce can visit all of the rooms by going through the following doorways (in order): 1, 3, 4, 5, 6, 7.

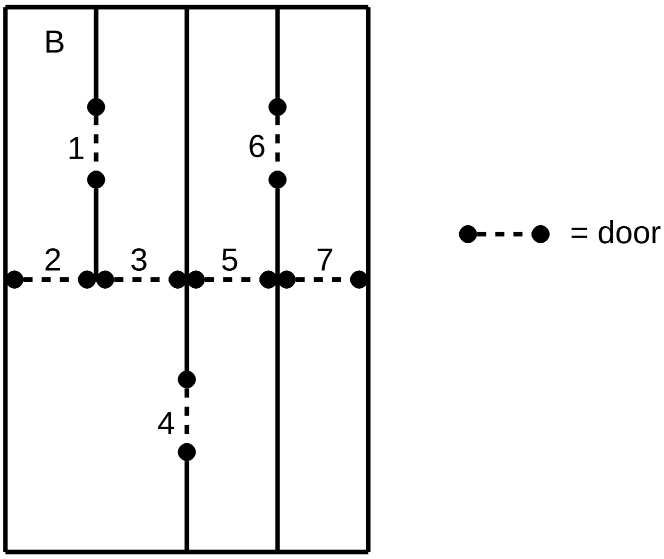

Figure 1: Example prison layout

Input (break.in)

The first line of input will contain three integers, N, M and B. N is the number of rooms in the cell-block (the rooms are numbered from 1 to N), M is the number of doorways in the cell-block, and B is the number of the room in which Bruce starts. The next M lines will describe the doorways: each of these lines will contain 2 integers, I and J, indicating that there is a doorway between the two rooms with numbers I and J.

Sample input

```
7 7 1
1 2
1 3
2 3
3 4
4 5
5 6
6 7
```

Output (break.out)

Your first line of output should contain a single integer, K, which is the maximum number of rooms that Bruce could visit (including the room that he starts in). On the next line you should then output $K-1$ integers describing the doorways (in order) that Bruce must pass through to visit this number of rooms (the doorways are numbered from 1 to M, and are given in this order in the input).

Sample output

```
7
1 3 4 5 6 7
```

Constraints

- $3 \leq N \leq 30$
- No room will have more than 4 doorways.

Scoring

If your output differs in any way from the format described above, then your program will score 0.

If the path you specify is invalid, you will score 0 points. If you have found a valid path for your value of K, then you will score $\frac{10}{1+(O-K)}$ where O is the actual maximum number of rooms that Bruce can visit. In the case that your score is not an integer, it will be rounded *downwards* to the nearest integer.

Store supplies

Author

Bruce Merry

Introduction

Fred the deranged store-keeper has become obsessed with ordering new supplies for his store, to the point where he has no time to actually sell anything. He is also obsessed with knowing how much of everything he has. Help him to manage his inventory.

Task

Everything that can be sold from Fred's store is identified by a positive integer. Whenever Fred orders things, he always orders one each of a contiguous range of identifiers (e.g. he will order one each of items 3, 4, 5 and 6). Initially Fred's store has no stock of anything. You will be given a list of orders that Fred makes, interspersed with queries about how many of a particular item there are. You must answer these queries before you will receive any more information about orders.

Example

Fred:

1. orders items 3–6;
2. orders items 5–10;
3. asks how many of item 6 there are;
4. asks how many of item 23 there are;
5. orders items 1–5.

The answers to his questions are 2 and 0 respectively.

Input

You must read input from standard input (as you would read the keyboard). Do not attempt to open an input file. There is no header, and each record in the file consists of one line, with either 2 or 3 space-separated integers number on it. The first number is either a 1 or 2. If it is a 1 then this is an order from Fred, and next two integers A and B satisfy $A < B$. Fred purchases one of everything from A to B inclusive. If the first number is a 2, then this is a query. The second number on the line is a product ID C, and you must say how many of item C are available. After the last record, there is a single line containing only the integer 0.

Input	Output
1 3 6	
1 5 10	
2 6	
	2
2 23	
	0
1 1 5	

Table 1: Sample interaction

Output

You must write your output to standard output (as you would write to screen). Do not attempt to open a file. For every query, you must output a single integer, the number of that item in stock. Remember that you will not receive the next record in the input until you have answered the query.

You must also flush the output stream to guarantee that the evaluator will receive the reply. This can be done as follows:

- Pascal: `flush(output);`
- C: `fflush(stdout);`
- C++: `cout.flush();`
- Java: `System.out.flush();`

In C++, using `endl` automatically flushes.

Testing

For testing purposes, you may upload an input file to the server. This file has the format described in the input section (i.e. the left hand column of table 1).

Constraints

- $1 \leq$ product ID ≤ 1000000.
- $1 \leq$ records in input ≤ 10000.

The record count does not include the terminating 0.

50% constraints

- $1 \leq$ product ID ≤ 1000.
- $1 \leq$ records in input ≤ 1000.

Time limit

2 seconds.

Scoring

Incomplete or invalidately formatted output scores 0. Otherwise, your score is the number of correct answers out of the total number of queries, with fractional scores rounded down.

Important note to Pascal users

Pascal users must *NOT* use the CRT unit. It can interfere with the input/output process. If you use the CRT unit and your program malfunctions, you will receive no compensation, even if it apparently worked when you wrote it.