**The 21st Tuymaada International Olympiad   
 Informatics, 2014**

**Day One**

1. A Numbers Game

|  |  |
| --- | --- |
| Input file name: | input.txt |
| Output file name: | output.txt |
| Time limit per test: | 1 sec |
| Memory limit: | 64 MiB |
| Points | 100 points |

Kolya and Vasya play a game. A natural number is randomly selected, and then each player must name a natural number not greater than . The player whose number has more divisors wins, and if the two numbers have the same number of divisors then the player with the smaller number wins. Note that the divisors of any number include both 1 and the number itself. Kolya badly wants to win this game and thinks that it would be great to have a program which will input and output the required number. Help him to write a program that will find the smallest number with the highest number of divisors.

Input format

The input file holds a single natural number ().

Output format

Output the smallest number with the most divisors. The number must not exceed .

Input and output examples

|  |  |
| --- | --- |
| input.txt | output.txt |
| 20 | 12 |
| 100 | 60 |
| 1024 | 840 |

1. Boolean equation system

|  |  |
| --- | --- |
| Input file name: | input.txt |
| Output file name: | output.txt |
| Time limit per test: | 1 sec |
| Memory limit: | 64 MiB |
| Points | 100 points |

How many solutions does the following system of equations have:

…

where () are Boolean variables?

Input format

The single line of the input file holds the number of equations in the system ().

Output format

Output a single number – the number of solutions for the system.

Input and output examples

|  |  |
| --- | --- |
| input.txt | output.txt |
| 2 | 10 |

1. **The** City **Day**

|  |  |
| --- | --- |
| Input file name: | input.txt |
| Output file name: | output.txt |
| Time limit per test: | 1 sec |
| Memory limit: | 64 MiB |
| Points | 100 points |

The city of Wonderight holds an annual festival, the City Day, on 15th of July. On this day one of the city squares is turned into a wonderland: a swimming pool, an ice rink (notwithstanding the July heat) and a lot of amusements and rides appear. All the kids living near the square, and all their parents, are absolutely happy that day. To please all the citizens, the Mayor has decided to change the festival square every year. Obviously, the square is closed to through traffic. However, Wonderight, like most modern cities, suffers from traffic jams. This year’s festival will be held at one of the squares that the Mayor travels through on his way to work. The Mayor can reach his office without traffic jams if the total number of ways to reach it is at least *N*. Help him to choose the festival square.

The map of Wonderight shows infinitely many vertical and infinitely many horizontal straight streets intersecting at right angles. Each intersection is a square with coordinates (, ), where and are the numbers of intersecting streets. The Mayor’s office is at the square with coordinates (0, 0). The Mayor always travels along a shortest route possible after the festival square has been closed to traffic.

(2,1)

(3,-2)

(0,0)

Input format

The input file consists of two lines, the first holding space-separated numbers *N* and *M*, and the second holding two numbers *k*, *l* being the coordinates of the square of the Mayor’s home (). The number *M* is either 1 or 2. If *M* = 1 then the number of routes from the Mayor’s home to the office must be exactly *N*, and if *M* = 2 then the number of possible routes should be at least *N*.

Output format

Output the number of the squares that can host the festival to the first line of the output file. Starting with the second line, output a pair of space-separated numbers to each line, the numbers being the coordinates of a possible square. If there is no square satisfying the restrictions, output 0.

Input and output examples

|  |  |
| --- | --- |
| input.txt | output.txt |
| 50 1  10 0 | 1  5 0 |
| 2 1  1 1 | 0 |
| 6 2  3 -2 | 6  0 -2  0 -1  1 -2  2 0  3 -1  3 0 |

1. Polylines

|  |  |
| --- | --- |
| Input file name: | input.txt |
| Output file name: | output.txt |
| Time limit per test: | 2 sec |
| Memory limit: | 64 MiB |
| Points | 100 points |

Consider an table where each cell holds four dots that are the vertices of a square joined either by two vertical lines (1st type) or by two horizontal lines (2nd type). It is known that the types of cells in the table alternate in a checkerboard fashion, with the upper left cell being of the 1st type. You need to supplement the system with horizontal of vertical unit line segments so that it will contain the minimum possible number of non-self-intersecting polylines.

Input format

The only line of the input file holds the number ().

Output format

Output the number of polylines to the first line. The following lines must each hold a number followed by pairs of numbers , where is the horizontal coordinate of a point and is the vertical coordinate. The lines connecting dots are taken to be of unit length so that the entire table has size . Coordinates of the upper left corner are , coordinates of the lower right corner are –. All the edges of the polyline must have unit length; however, an edge can be a direct continuation of another one.

Input and output examples

|  |  |  |
| --- | --- | --- |
| input.txt | output.txt | *figure* |
| 1 | 1  3 0 0 1 0 1 1 0 1 |  |
| 2 | 2  7 0 0 0 1 1 1 1 0 2 0 3 0 3 1 2 1  7 1 2 0 2 0 3 1 3 2 3 2 2 3 2 3 3 |  |

Comments

The initial connections are shown in gray in the figures. A possible solution shows its additional lines in black

1. Zombies, zombies, zombies

|  |  |
| --- | --- |
| Input file name: | input.txt |
| Output file name: | output.txt |
| Time limit per test: | 3 sec |
| Memory limit: | 64 MiB |
| Points | 100 points |

Vitya has at last managed to buy the *Gnomes vs Zombies* game. Similar to another well-known game, the zombies are attacking your house from the backyard. The backyard is an rectangle, and your house is adjacent to its right-hand side. The zombies may appear in any square, but after that they only move horizontally to the right. A square can hold at most one zombie.

Some squares also have garden gnomes, it is with them that the player must atop the hordes of zombies. At the beginning of the game the gnomes lie underground, but when a zombie steps into the square the player may choose to activate the gnome, so that the gnome will move to the surface and attack the zombie. During an attack the gnome and the zombie destroy one another. If any zombie reaches the house, you lose. The zombie is considered to have reached your house if it moves to the right while standing at a rightmost cell of the backyard.

After a couple of hours Vitya has completed the game, and it seemed too easy to him. So Vitya has decided to change some things and hacked the game. Now zombies can move vertically by a certain number of squares.

Unfortunately after this innovation some levels have become impassable. Vitya managed to avoid this by introducing another novelty. The house has a lot of stuff to make traps with, so you can allow some zombies to reach the house, where you will trap and destroy them inside. A trap is only good for one zombie. However, Vitya has failed to calculate the precise number of traps needed to complete the level (allowing too many traps would make the level easy and boring). So Vitya has asked you to write a program that, given the level data (the size of the backyard, the location of the garden gnomes, the location and movement of zombies), will calculate the minimum number of zombies that will reach the house after an optimal play.

Input format

The first line of the input file holds the numbers (), () and (). These are the dimensions of the backyard, being the number of columns, and  is the number of gnomes. Then lines follow, each holding a pair of integer numbers ,  – the numbers of the rows and columns where the gnomes lie. The remaining lines hold zombie commands. These come in three flavours:

1. "brains!!! i j" orders a zombie to appear in a certain square.

2. "brains### i j s" orders a zombie to move vertically by squares, so that the zombie moves to the square with coordinates . Unfortunately, Vitya has not managed to write the animation code, so the movement actually looks like teleportation, which means the zombie cannot be destroyed in transit until it reaches the destination square of .

3. "brains..." makes all the zombies already in the backyard to move one step to the right.

Each command is on its own line, and the total number of commands does not exceed 200000. The last line holds a string "brains???" which means that the morning has come and all the zombies in the backyard have turned to ashes.

It is guaranteed that the zombies do not leave the backyard rectangle when moving vertically, and also the number of zombies is no greater than the number of gnomes.

Output format

Output a single number, the minimum number of the zombies that will reach the house during an optimal play.

Input and output examples

|  |  |
| --- | --- |
| input.txt | output.txt |
| 3 3 1  3 2  brains!!! 1 1  brains...  brains...  brains...  brains??? | 1 |
| 3 3 2  1 2  2 2  brains!!! 2 2  brains!!! 2 1  brains### 2 2 -1  brains...  brains### 2 2 1  brains...  brains...  brains??? | 0 |
| 3 3 4  1 1  1 3  3 1  3 3  brains!!! 2 1  brains...  brains### 2 2 1  brains### 3 2 -2  brains!!! 2 2  brains!!! 1 1  brains...  brains!!! 2 1  brains...  brains...  brains??? | 1 |