NOI '14 P3 - Deletion Game

Time limit: 4.5s **Memory limit:** 512M

National Olympiad in Informatics, China, 2014

Recently, little Z has been addicted to a new type of deletion game. This game takes place on an $n \times m$ grid. Initially, the grid cells are filled with integers from 0 to 9. Through deletion, the cells become blank (denoted by -1). We will denote the cell at row i, column j as $A_{i,j}$, and let its coordinates be (i, j).

Given three parameters $l_{\min}$, $l_{\max}$, and K, the player can make **no greater** than K moves. For each move, the player must first find a path of length l in the grid. Formally, this path is represented by two length l sequences $x_1, x_2, \ldots, x_l$ and $y_1, y_2, \ldots, y_l$ that satisfy the following conditions:

1. $1 \le x_i \le n$ and $1 \le y_i \le m$, where $1 \le i \le l$, meaning that (x_i, y_i) is a valid cell on the grid.
2. $|x_i - x_{i+1}| + |y_i - y_{i+1}| = 1$ where $1 \le i < l$, meaning that (x_i, y_i) and (x_{i+1}, y_{i+1}) are two neighbouring cells on the grid.
3. $x_i \ne x_j$ or $y_i \ne y_j$, where $1 \le i < j \le l$, meaning that the path does not repeat cells.
4. $A_{x_i,y_i} \ne -1$, where $1 \le i \le l$, meaning that the path does not go through any blank cells.
5. $A_{x_1,y_1} \ne 0$, meaning that the path does not start with the number 0.
6. $l_{\min} \le l \le l_{\max}$, meaning that the length of the path has to fall within the given range.

Then, the player will string the digits on the path together into an integer N. Formally:

$$N = \sum_{i=1}^{l} A_{x_i,y_i} \times 10^{l-i}$$

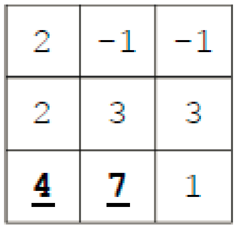
The game will provide two parameters c_1 and c_2 to calculate the player's score on a move:

1. If N is a prime number, then the **prime subscore** is l^{c_1}, otherwise the **prime subscore** is 1.
2. If N is a palindromic number (i.e. when viewed as a string, N is identical whether its letters are arranged forwards or reverse), then the **palindromic subscore** is l^{c_2}, otherwise the **palindromic subscore** is 1.
3. If the prime subscore and the palindromic subscore are both equal to 1, then the overall score for the move is 0. Otherwise, the overall score for the move is equal to the sum of the prime subscore and the palindromic subscore.

If after a move the move's overall score is 0, then you will have wasted the move and the game's state will not change. Otherwise, if the move's overall score is greater than 0, then all cells on the path will be set to blank, and the cells above the blank will drop down. Formally, the following operations take place:

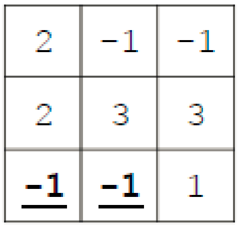
1. Set $A_{x_i,y_i} \leftarrow -1$, where $1 \le i \le l$.
2. Examine every cell in the grid. If a cell (i, j) satisfies $i \ne n$, $A_{i,j} \ne -1$, and $A_{i+1,j} = -1$, then set $A_{i+1,j} \leftarrow A_{i,j}$ and $A_{i,j} \leftarrow -1$. Repeat this step until no more cells in the grid satisfy the conditions.

The following is an example of a player making a move and deleting some cells. In this case, $n = m = 3$ and $c_1 = c_2 = 1$. The player starts off facing the board in figure 1.



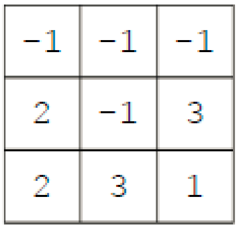
2	-1	-1
2	3	3
4	**7**	1

Fig. 1



2	-1	-1
2	3	3
-1	**-1**	1

Fig. 2



-1	-1	-1
2	-1	3
2	3	1

Fig. 3

1. The player selects the path containing digits 4 and 7, which strung together makes $N = 47$. Since 47 is a prime number, the prime subscore is $l^{c_1} = 2^1 = 2$. Since 47 is not a palindromic number, the palindromic subscore is 1. Thus, the overall score for the move is $2 + 1 = 3$.
2. Since the score of the move is nonzero, the path's values are deleted (as in figure 2).
3. Digits above the blank cells drop down (as in figure 3). At this point, the player is ready to make the next move.

We will also give you a parameter F. After the player has made every move, the player's **final score** will be determined using F. If $F = 0$, then the player's final score is equal to the sum of the scores across every move. If $F = 1$, then the player's final score will be sum of the scores across every move, divided by 2^d, rounded down. Formally:

$$\text{Final score} = \begin{cases} \text{Sum of scores across all moves}, & F = 0 \\ \left\lfloor \frac{\text{Sum of scores across all moves}}{2^d} \right\rfloor, & F = 1 \end{cases}$$

where d is the final number of non-blank cells at the end of the game.

Little Z finds himself hooked to this game, unable to step away. She would like your help. For the given parameters, provide a series of moves to play the game. Of course, the final score should be as high as possible.

Input Specification

There will be 10 files `game1.in` ~ `game10.in` that will be given to your program (through standard input). They can be downloaded here for you to study: game.zip.

For each test case:

The first line of input will contain an integer from 1 to 10, representing the test case number. Test case `gamei.in` will have i on its first line.
The second line of input will contain eight space-separated integers n, m, K, l_{min}, l_{max}, c_1, c_2, and F.
The next n lines will each contain m integers, representing the array A. Adjacent integers on the same line will be separated by a space.
The input will not contain any extra or trailing spaces.

Output Specification

The first line of output should contain a single integer M $(0 \leq M \leq K)$, representing the number of moves you would like to make.
M lines should follow, each describing one move. For each of these lines, the first integer should be l, representing the length of the path. $2l$ integers should follow on the same line. They are, respectively, $x_1, y_1, x_2, y_2, \ldots, x_l, y_l$.
The output should not contain extra spaces or lines. Adjacent values on the same line should be separated by a single space.
Your output should not exceed 1MB in size. The data guarantees that a valid answer will not exceed this upper bound.

Sample Input 1

```
0
3 3 100 2 3 1 1 0
2 1 1
2 3 3
4 7 1
```

Sample Output 1

```
4
2 2 2 3 2
2 3 1 3 2
2 2 1 3 1
3 1 3 2 3 3 3
```

Explanation 1

Four deletions are made. The deleted paths and move scores are, respectively: 37 (for a score of $2+1=3$), 41 (for a score of $2+1=3$), 22 (for a score of $1+2=3$), and 131 (for a score of $3+3=6$). Since $F=0$, the final score is simply 15. There may exist a more optimal solution.

Sample Input 2

```
0
1 3 100 2 3 1 1 1
2 1 1
```

Sample Output 2

```
1
2 1 2 1 3
```

Explanation 2

Only one deletion is made. The deleted path is 11. The score is $2+2=4$. Since $F=1$, the final score is equal to 4, the sum of scores across all moves, divided by $2^1=2$, rounded down, which is 2. If the deleted path was 211, then an optimal final score of 4 would have been obtained.

Grading

For each test case, we have set up 9 parameters $a_{10}, a_9, \ldots, a_2$. If your output is invalid, then your score will be 0 for the test case. Otherwise, assuming your final score is w_{user}, then your score out of 10 for the test case will be given by the following table.

Score	Condition
10	$w_{\text{user}} \geq a_{10}$
9	$w_{\text{user}} \geq a_9$
8	$w_{\text{user}} \geq a_8$
7	$w_{\text{user}} \geq a_7$
6	$w_{\text{user}} \geq a_6$
5	$w_{\text{user}} \geq a_5$
4	$w_{\text{user}} \geq a_4$
3	$w_{\text{user}} \geq a_3$
2	$w_{\text{user}} \geq a_2$
1	$w_{\text{user}} > 0$

Experimentation

We provide a tool `check` (GNU/Linux: check, Windows: check.exe) for you to help check if your output program is valid.

The usage for this program (on Linux) is

```
./check <case_no>
```

where `<case_no>` is the number of the test case.

On Windows, use `.\` instead of `./`.

The checker should be placed in the same directory as your input and output files.

For example, `./check 3` should be executed in a directory containing files `game3.in` and `game3.out` to test whether the output would be accepted.

After you invoke this program, the checker will provide the result to the execution of your output file in one of the following ways:

- `Abnormal termination`: An unknown error occurred.
- `Input/Output file does not exist.`: We cannot load your input or output file.
- `Output invalid!`: There is an error with your output file. A general error message may now be included.
- `Details: xxx.`: Other information.
- `Correct! Your answer is x.`: Your output is acceptable. Your final score is x.

Note that the checker is not guaranteed to work for inputs other than the ones provided.

Problem translated to English by **Alex**.