Navigation 2

JOI Kingdom is an island surrounded by sea. The land of JOI Kingdom is a grid of square cells consisting of N rows and N columns. The vertical direction is the north-south direction, and the horizontal direction is the west-east direction. The cell in the $(r + 1)$-th row $(0 \leq r \leq N - 1)$ from north and the $(c + 1)$-th column $(0 \leq c \leq N - 1)$ from west is denoted by the cell (r, c).

Anna, the queen of JOI Kingdom, will invite Bruno to a party. She is now choosing the place for the party. She already selected K $(= 7)$ candidate cells of the place for the party. The candidate cells are numbered from 0 through $K - 1$. The candidate cell i is the cell (R_i, C_i). No candidate cell is adjacent to sea.

The place for the party will be determined on the day of the party.

On the day before the party, in order to help Bruno arrive at the place for the party without getting lost, Anna will place a flag in every cell. On each flag, Anna will write an integer between 1 and 1 000 000 000, inclusive.

On the day of the party, only the index t $(0 \leq t \leq K - 1)$ of the candidate cell will be told to Bruno. After that, Bruno will take a helicopter and arrive at a cell which is not adjacent to sea. Then Bruno will start moving to the place for the party.

Bruno does not know where he is, but he knows the directions of north, south, east, and west. Bruno can only see a flag in his current cell and its 8 surrounding cells. In other words, when Bruno is at the cell (a, b) $(1 \leq a \leq N - 2, 1 \leq b \leq N - 2)$, he can only see the flags in the following 9 cells:

$$(a-1, b-1),\ (a-1, b),\ (a-1, b+1),\ (a, b-1),\ (a, b),\ (a, b+1),\ (a+1, b-1),\ (a+1, b),\ (a+1, b+1)$$

Bruno can take one of the following 5 actions.

- Action 0: Bruno moves one cell to the east. Namely, he moves from the cell (a, b) to the cell $(a, b + 1)$.
- Action 1: Bruno moves one cell to the west. Namely, he moves from the cell (a, b) to the cell $(a, b - 1)$.
- Action 2: Bruno moves one cell to the south. Namely, he moves from the cell (a, b) to the cell $(a + 1, b)$.
- Action 3: Bruno moves one cell to the north. Namely, he moves from the cell (a, b) to the cell $(a - 1, b)$.
- Action 4: Bruno thinks the party will be held at his current cell, and stays there. He stops moving.

Since it is forbidden to arrive at the party late, Bruno has to move to the place for the party so that the number of actions is the minimum possible. Thus, by the setting of this task, Bruno will never enter into a cell adjacent to sea.

As it is tiresome to write large integers on flags, Anna wants to minimize the maximum integer written on the flags.

Write a program which implements Anna's strategy and Bruno's strategy. Anna will write integers on the flags, and Bruno has to arrive at the place for the party with minimum number of actions.

Implementation Details

You need to submit two files.

The first file is `Anna.cpp`. It should implement Anna's strategy. It should implement the following function. The program should include `Anna.h` using the preprocessing directive `#include`.

- `void Anna(int N, int K, std::vector<int> R, std::vector<int> C)`
  This function implements Anna's strategy to write integers on the flags. For each scenario (see **Scoring**), this function is called exactly once in the beginning.
  - The parameter `N` means the land of JOI Kingdom is a grid of square cells consisting of N rows and N columns.
  - The parameter `K` is the number K ($= 7$) of the candidate cells of the place for the party.
  - The parameters `R` and `C` are arrays of length K. Here `R[i]` and `C[i]` ($0 \leq i \leq K-1$) denote the cell (R_i, C_i) of the candidate cell i.
  - For the range of the values of the parameters `N`, `K`, `R[i]` and `C[i]` ($0 \leq i \leq K-1$) see **Constraints**.

For each function call to `Anna`, the following function should be called exactly N^2 times. It should be called once for every cell.

- `void SetFlag(int r, int c, int value)`
  - The parameters `r` and `c` mean Anna writes an integer on the flag at the cell (`r`, `c`). Here $0 \leq$ `r` $\leq N-1$, $0 \leq$ `c` $\leq N-1$ should be satisfied. If this condition is not satisfied, your program is judged as **Wrong Answer [1]**.
  - The parameter `value` is the integer Anna writes on the flag. Here $1 \leq$ `value` $\leq 1\,000\,000\,000$ should be satisfied. If this condition is not satisfied, your program is judged as **Wrong Answer [2]**.
  - If the function `SetFlag` is called with the same parameters (`r`, `c`) more than once, your program is judged as **Wrong Answer [3]**.
  - When the function `Anna` terminates, if the number of function calls to the function `SetFlag` is different from N^2, your program is judged as **Wrong Answer [4]**.

When a function call to the function `SetFlag` is considered as a Wrong Answer, your program is terminated immediately.

The second file is `Bruno.cpp`. It should implement Bruno's strategy. It should implement the following function. The program should include `Bruno.h` using the preprocessing directive `#include`.

- `std::vector<int> Bruno(int K, std::vector<int> value)`
  This function should describe Bruno's next action. For each scenario (see **Scoring**), after the function `Anna` is called, this function is called **exactly once**.
  - The parameter `K` is the number of candidate cells $K(= 7)$.
  - The parameter `value` is an array of length 9. It contains the integers written on the flags at Bruno's current cell and its 8 surrounding cells. More precisely, if Bruno is currently at the cell (a, b) $(1 \leq a \leq N-2,\ 1 \leq b \leq N-2)$, the values of `value[0]`, `value[1]`, ..., `value[8]` are the integers written on the flags in the cells

    $(a-1, b-1)$, $(a-1, b)$, $(a-1, b+1)$, $(a, b-1)$, (a, b), $(a, b+1)$, $(a+1, b-1)$, $(a+1, b)$, $(a+1, b+1)$,

    respectively.
  - For every $t = 0, 1, 2, \ldots, K-1$, the function `Bruno` should determine Bruno's next action when the party will be held at the candidate cell t. The return value is an array of length K. The $(i+1)$-th element $(0 \leq i \leq K-1)$ of the array should be Bruno's next action for $t = i$.
  - If the return value is not an array of length K, your program is judged as **Wrong Answer [5]**.
  - Every element in the array should be one of 0, 1, 2, 3, or 4. If this condition is not satisfied, your program is judged as **Wrong Answer [6]**.
  - For every t, the action given by the function `Bruno` should be Bruno's next action so that he will move to the place of the party with minimum number of actions. In particular, if his next action is Action 4, his current cell should be the place for the party. If this condition is not satisfied, your program is judged as **Wrong Answer [7]**. If there are multiple ways to move to the place for the party with the minimum number of actions, the return value can be any one of them.

In this task, each test case consists of Q scenarios. For each scenario, each of the function `Anna` and the function `Bruno` is called exactly once. Therefore, for each test case, each of the function `Anna` and the function `Bruno` is called Q times. These functions are called alternately. See **Scoring** for details.

Important Notices

- Your program can implement other functions for internal use, or use global variables. Submitted files will be compiled with the grader, and become a single executable file. All global variables and internal functions should be declared in an unnamed namespace to avoid confliction with other files. When it is graded, it will be executed as two processes of Anna and Bruno. The process of Anna and the process of Bruno cannot share global variables.
- Your program must not use the standard input and the standard output. Your program must not communicate with other files by any methods. However, your program may output debugging information to the standard error.

Scoring

Each test case consists of Q scenarios. The scenarios are numbered from 0 through $Q-1$. For each scenario, the following values are fixed. For the range of these values, see **Constraints**.

- The vertical and the horizontal size N of JOI Kingdom.
- The number of candidate cells $K(=7)$.
- The candidate cells $(R_0, C_0), (R_1, C_1), \ldots, (R_{K-1}, C_{K-1})$ of the place for the party.
- Bruno's current cell (a, b).

The function `Anna` is called for each scenario. For given parameters, Anna should write integers on flags. The function `Bruno` is also called for each scenario. It should determine Bruno's next action. The function `Anna` and the function `Bruno` are called in the following way.

1. For each $k = 0, 1, 2, \ldots, Q-1$, in order, the following 2. and 3. are performed, in this order.
2. The function `Anna` is called. The parameters for the scenario k are given as in **Implementation Details**.
3. The function `Bruno` is called. The parameters for the scenario k are given as in **Implementation Details**.

If your program is judged as Wrong Answer during this process, your program is terminated immediately, and it is considered as a Wrong Answer for the test case.

Compilation and Test Run

You can download an archive file from the contest webpage which contains the sample grader to test your program. The archive file also contains a sample source file of your program.

The sample grader is the file `grader.cpp`. In order to test your program, put `grader.cpp`, `Anna.cpp`, `Bruno.cpp`, `Anna.h`, `Bruno.h` in the same directory, and run the following command to compile your programs.

```
g++ -std=gnu++17 -O2 -fsigned-char -o grader grader.cpp Anna.cpp Bruno.cpp
```

When the compilation succeeds, the executable file `grader` is generated.

Note that the actual grader is different from the sample grader. The sample grader will be executed as a single process, which will read input data from the standard input and write the results to the standard output.

Input for the Sample Grader

The sample grader reads the following data from the standard input. Given values are all integers.

Q
(Input for scenario 0)
$\vdots$
(Input for scenario $Q-1$)

The input for each scenario is given as follows.

$N\ K$
$R_0\ C_0$
$\vdots$
$R_{K-1}\ C_{K-1}$
$a\ b$

As the input for the sample grader, you can set the value of N in the range $3 \leq N \leq 100$, and the value of K in the range $1 \leq K \leq 7$. Note that these ranges are different from the actual constraints of this problem.

Output of the Sample Grader

When the program terminates successfully, the sample grader writes the following information to the standard output (quotes for clarity).

- If the answer is correct, it writes the maximum number written on the flags by the function `Anna` as "`Accepted : Maximum value = 12`".
- If your program is judged as Wrong Answer, it writes its type as "`Wrong Answer [1]`".

If your program is judged as multiple types of Wrong Answer, the sample grader reports only one of them.

Constraints

- $1 \leq Q \leq 300$.
- $5 \leq N \leq 100$.
- $K = 7$.
- $1 \leq R_i \leq N - 2$ $(0 \leq i \leq K - 1)$.
- $1 \leq C_i \leq N - 2$ $(0 \leq i \leq K - 1)$.
- $(R_i, C_i) \neq (R_j, C_j)$ $(0 \leq i < j \leq K - 1)$.
- $1 \leq a \leq N - 2$.
- $1 \leq b \leq N - 2$.

Grading

If your program is judged as Wrong Answer in any one of the test cases, your score for this task is 0 point.

If your program is judged as correct for every test case, your score for this task is calculated as follows. Let L be the maximum integer written on the flags among all test cases.

- If $70\,001 \leq L \leq 1\,000\,000\,000$, your score is 7 points.
- If $10\,001 \leq L \leq 70\,000$, your score is 13 points.
- If $2\,001 \leq L \leq 10\,000$, your score is 19 points.
- If $21 \leq L \leq 2\,000$, your score is $50 - 12.5 \times \log_{10}\left(\frac{L}{20}\right)$ points, rounded down to the nearest integer.

If $L \leq 20$, your score is given by the following table.

L	20	19	18	17	16	15	14	13	≤ 12
Score	50	53	56	60	64	69	75	85	100

Sample Communication

Here is a sample input for the sample grader and corresponding function calls. In the following example, the integers written by Anna on the 25 flags are as in the following table.

Sample Input 1	Integers written by Anna on the flags
1	47 15 63 56 71
5 7	10 46 52 18 67
1 1	63 56 71 19 48
1 2	52 18 67 99 26
2 1	71 19 48 60 89
2 2	
2 3	
3 2	
3 3	
1 1	

Call to Anna	Call to Bruno	Return value
`Anna(5,7,[1,1,2,...,3],[1,2,1,...,3])`		
`SetFlag(0,0,47)`		
`SetFlag(0,1,15)`		
`SetFlag(0,2,63)`		
⋮		
`SetFlag(4,4,89)`		
	`Bruno(7,[47,15,63,...,71])`	`[4,0,2,2,2,0,0]`

In this sample input, $(a, b) = (1, 1)$. When the party is held at the candidate cell 0, 1, 2, or 3, Bruno's next action should be as follows, and the function `Bruno` should return the actions accordingly.

- If the candidate 0 is chosen, the place of the party will be the cell $(1, 1)$, and Bruno has to take Action 4.
- If the candidate 1 is chosen, the place of the party will be the cell $(1, 2)$, and Bruno has to take Action 0.
- If the candidate 2 is chosen, the place of the party will be the cell $(2, 1)$, and Bruno has to take Action 2.
- If the candidate 3 is chosen, the place of the party will be the cell $(2, 2)$, and Bruno has to take either Action 0 or Action 2.

In this sample communication, the return values are `[4,0,2,2,2,0,0]`. There can be multiple ways to move to the place for the party with the minimum number of actions. For example, your program is judged as correct as well if the return values are `[4,0,2,0,2,0,2]`.

Sample Input 2

```
1
100 7
3 21
16 9
44 36
44 78
45 78
67 59
90 22
84 59
```

In this sample input, your program is judged as correct if the return values of the function `Bruno` are `[3,1,1,0,0,3,2]`, for example.