Stray Cat

Anthony is an ant living in JOI City. There are N towns in JOI City, numbered from 0 to $N-1$. Anthony lives in the town 0. There are M roads, numbered from 0 to $M-1$. The road i ($0 \leq i \leq M-1$) connects the town U_i and the town V_i, and it is possible to pass through it in both directions. Different roads connect different pairs of towns. It is possible to travel from any town to any other town by passing through several roads.

Catherine is a cat who is a friend of Anthony. She is planning to visit JOI City, but she does not know the information of the roads and she often strays. Anthony decided to put marks on the roads in advance. There are A types for marks, numbered from 0 to $A-1$.

Now Catherine arrived at a town in JOI City. Whenever she is in a town other than the town 0, she does the following:

> For each type of marks, she can count the number of roads of that type going from her current town, except for the road which she passed lastly (if such a road exists).

After that, she chooses a road to pass. Note that **except for the road which she passed lastly, she can distinguish the road only by types of marks.** Choosing roads suitably, she wants to arrive at the town 0 without taking much time. More precisely, the minimum number of roads to pass to travel from her first town to the town 0 is d, she wants to arrive at the town 0 by choosing roads at most $d + B$ times.

Write a program which, given the information of the roads, implements Anthony's strategy to put marks on the roads, and a program which implements Catherine's strategy to choose roads.

Implementation Details

You need to submit two files.

The first file is `Anthony.cpp`. It implements Anthony's strategy and it should implement the following function. The program should include `Anthony.h`.

- `std::vector<int> Mark(int N, int M, int A, int B,`
  `std::vector<int> U, std::vector<int> V)`

  This function is called exactly once in the beginning.
  - The parameter `N` is the number of towns N.
  - The parameter `M` is the number of roads M.
  - The parameter `A` is the number of types of marks A.
  - The parameter `B` is the margin for the number of times Catherine can choose roads.

- The parameters `U` and `V` are arrays of length M, where `U[i]` and `V[i]` are the towns U_i and V_i connected by the road i ($0 \leq i \leq M-1$).
- The return value `x` should be an array of length M. If the length is different from M, your program is judged as **Wrong Answer [1]**. The value `x[i]` ($0 \leq i \leq M-1$) represents the mark put on the road i. The inequality $0 \leq$ `x[i]` $\leq A-1$ should be satisfied. If $0 \leq$ `x[i]` $\leq A-1$ is not satisfied, your program is judged as **Wrong Answer [2]**.

The second file is `Catherine.cpp`. It implements Catherine's strategy and it should implement the following function. The program should include `Catherine.h`.

- `void Init(int A, int B)`
  This function is called exactly once in the beginning.
  - The parameter `A` is the number of types of marks A.
  - The parameter `B` is the margin for the number of times Catherine can choose roads.
- `int Move(std::vector<int> y)`
  This function is called whenever Catherine arrives at a town different from the town 0.
  - The parameter `y` is an array of length A such that `y[j]` is the number of roads from her current town whose mark is j ($0 \leq j \leq A-1$), except for the road she passed lastly (if such a road exists).
  - The return value `z` should satisfy $-1 \leq$ `z` $\leq A-1$. If $-1 \leq$ `z` $\leq A-1$ is not satisfied, your program is judged as **Wrong Answer [3]**. If `z` $= -1$, she turns back choosing the road she passed lastly. If $0 \leq$ `z` $\leq A-1$, she chooses a road whose mark is `z`. When the function `Move` is called for the first time, if `z` $= -1$, then your program is judged as **Wrong Answer [4]**. If $0 \leq$ `z` $\leq A-1$ and `y[z]` $= 0$, your program is judged as **Wrong Answer [5]**.

If Catherine chooses a road which is different from the road she passed lastly, the road she will actually pass will be one of the road whose mark is specified by the return value of the function `Move`. **Note that this choice may not be made at random.**

If Catherine does not arrive at the town 0 after she passes roads $d+B$ times (i.e., if the function `Move` is called $d+B$ times), your program is judged as **Wrong Answer [6]**.

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 Anthony and Catherine. The process of Anthony and the

process of Catherine 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.

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`, `Anthony.cpp`, `Catherine.cpp`, `Anthony.h`, `Catherine.h` in the same directory, and run the following command to compile your programs.

```
g++ -std=gnu++14 -O2 -o grader grader.cpp Anthony.cpp Catherine.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.

$N\ M\ A\ B\ S$
$U_0\ V_0$
$\vdots$
$U_{M-1}\ V_{M-1}$

Here S is the index of the town where Catherine arrived at first.

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 your program is judged as Wrong Answer [1], [2], [3], [4] or [5], it writes its type as "`Wrong Answer [1]`".

- If Catherine does not arrive at the town 0 after $N + B$ moves, it writes "`Wrong Answer; Number of moves > N + B`".
- Otherwise, it writes the number of moves (the number of calls to the function `Move`) as "`Number of moves = 4`". Note that it does not judge whether it is correct or Wrong Answer [6].

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

In the sample grader, when Catherine chooses a road which is different from the road she passed lastly, the road she will actually pass is chosen uniformly at random among the roads with the mark specified by the return value of the function `Move`, using pseudorandom numbers with a fixed seed. In order to run the program with a different seed, execute the grader as follows:

```
./grader 2020
```

Here the first argument is an integer which is the seed of the pseudorandom numbers.

Constraints

- $2 \leq N \leq 20\,000$.
- $1 \leq M \leq 20\,000$.
- $1 \leq S \leq N - 1$ (S is the number of the town where Catherine arrived at first).
- $0 \leq U_i < V_i \leq N - 1$ $(0 \leq i \leq M - 1)$.
- $(U_i, V_i) \neq (U_j, V_j)$ $(0 \leq i < j \leq M - 1)$.
- It is possible to travel from any town to any other town by passing through several roads.

Subtasks

1. (2 points) $A = 4,\ B = 0,\ M = N - 1$.
2. (2 points) $A = 4,\ B = 0$.
3. (2 points) $A = 3,\ B = 0,\ M = N - 1$.
4. (9 points) $A = 3,\ B = 0$.
5. (5 points) $A = 2,\ B = 2N,\ M = N - 1,\ 6 \leq N \leq 500$.
6. (71 points) $A = 2,\ B = 12,\ M = N - 1$.
7. (9 points) $A = 2,\ B = 6,\ M = N - 1$.

Sample Communication

Here is a sample input for the sample grader and corresponding function calls.

Sample Input 1

`7 6 2 6 1`
`0 2`
`0 4`
`1 2`
`1 3`
`1 5`
`4 6`

Anthony		Catherine	
Call	Return	Call	Return
`Mark(7,6,2,6,[0,0,1,1,1,4],[2,4,2,3,5,6])`	`[1,0,0,1,0,1]`		
		`Init(2,6)`	
		`Move([2,1])`	`0`
		`Move([0,0])`	`-1`
		`Move([1,1])`	`0`
		`Move([0,1])`	`1`

In this sample communication, Catherine visits the town 1, 5, 1, 2, 0, in this order. We have $d = 2$. Catherine moves 4 times.

This sample input satisfies the constraints for Subtask 7.

Among the files you can download from the contest site, `sample-02.txt` satisfies the constraints for Subtask 4.