Airline Route Map

Alice is living in JOI Kingdom. She will invite Bob, who is living in Republic of IOI. Before she invites him, she is planning to send the airline route map of JOI Kingdom to him. JOI Kingdom is an island country consisting of N islands, numbered from 0 to $N-1$. There are M airline routes in JOI Kingdom. For each i $(0 \le i \le M-1)$, the $(i+1)$-th airline route connects the island A_i and the island B_i, in both directions. No two airline routes connect the same two islands. She must use a special telegraph machine operated by JOI Kingdom. She can send an undirected graph using the telegraph machine. However, when she uses it, the numbers of the vertices and the number of the edges will be shuffled randomly.

Precisely, the information will be sent as follows. Let G be the graph sent by Alice. (Let V be the number of vertices of G, and U the number of edges of G.)

- Alice specifies the number of edges V of G, and the number of edges U of G. Then, she puts each of numbers $0, 1, \dots, V-1$ to each vertex, and each of numbers $0, 1, \dots, U-1$ to each edge.
- Alice specifies the parameters $C_0, C_1, \dots, C_{U-1}$ and $D_0, D_1, \dots, D_{U-1}$. These parameters describe the edges of G, i.e., for each j $(0 \le j \le U-1)$, the j-th edge of G connects the vertex C_j and the vertex D_j.
- The numbers of the vertices of G are shuffled by JOI Kingdom. First, JOI Kingdom generates a sequence $p[0], p[1], \dots, p[V-1]$, which is a permutation of $0, 1, \dots, V-1$. Then, $C_0, C_1, \dots, C_{U-1}$ are replaced by $p[C_0], p[C_1], \dots, p[C_{U-1}]$, and $D_0, D_1, \dots, D_{U-1}$ are replaced by $p[D_0], p[D_1], \dots, p[D_{U-1}]$.
- Then, the numbers of the edges of G are shuffled by JOI Kingdom. First, JOI Kingdom generates a sequence $q[0], q[1], \dots, q[U-1]$ which is a permutation of $0, 1, \dots, U-1$. Then, $C_0, C_1, \dots, C_{U-1}$ are replaced by $C_{q[0]}, C_{q[1]}, \dots, C_{q[U-1]}$, and $D_0, D_1, \dots, D_{U-1}$ are replaced by $D_{q[0]}, D_{q[1]}, \dots, D_{q[U-1]}$.
- The following data are sent to Bob: the values of V and U, and the latest values of the parameters $C_0, C_1, \dots, C_{U-1}$ and $D_0, D_1, \dots, D_{U-1}$.

Note that only a simple graph can be sent using this telegraph machine. Here, a simple graph means a graph without multiple edges and self-loops.

In other words, she can send a graph satisfying the following conditions: $(C_i, D_i) \ne (C_j, D_j)$ and $(C_i, D_i) \ne (D_j, C_j)$ are satisfied for every i, j $(0 \le i < j \le U-1)$, and $C_i \ne D_i$ is satisfied for every i $(0 \le i \le U-1)$.

Alice wants to send the airline route map of JOI Kingdom to Bob using a graph with minimum number of vertices.

Task

In order to help communication between Alice and Bob, write the following two programs:

- Given the number of islands N in JOI Kingdom, the number of airline routes M in JOI Kingdom, and the sequences A, B representing the airline route map of JOI Kingdom, the first program outputs information of the graph G sent by Alice.
- Given information of the graph G received by Bob, the second program recovers the airline route map of JOI Kingdom.

Implementation Details

You need to submit two files.

The first file is `Alice.cpp`. This file outputs information of the graph sent by Alice. It should implement the following function. The program should include `Alicelib.h`.

- `void Alice( int N, int M, int A[], int B[] )`

  For each test case, this function is called once.

  - The parameter `N` is the number of islands of JOI Kingdom.
  - The parameter `M` is the number of airline routes in JOI Kingdom.
  - The parameters `A[]`, `B[]` are sequences of length M describing the airline route map of JOI Kingdom.

  Using the following functions, the function `Alice` outputs information of the graph G sent by Alice.

  ★ `void InitG( int V, int U )`

  This function specifies the number of vertices of G and the number of edges of G.

  - ◇ The parameter `V` is the number of vertices of G. The parameter `V` should be an integer between 1 and 1500, inclusive. If the call to this function has parameters outside this range, your program is considered as **Wrong Answer [1]**.
  - ◇ The parameter `U` is the number of edges of G. The parameter `U` should be an integer between 0 and $V(V-1)/2$, inclusive. If the call to this function has parameters outside this range, your program is considered as **Wrong Answer [2]**.

  ★ `void MakeG( int pos, int C, int D )`

  This function specifies the edges of G.

  - ◇ The parameter `pos` is the number of the edge specified by the call. The parameter `pos` should be an integer between 0 and $U-1$, inclusive. If the call to this function has parameters outside this range, your program is considered as **Wrong Answer [3]**. This function should not be called more than once with the same parameter `pos`. If this function is called more than once with the same parameter, your program is considered as **Wrong Answer [4]**.
  - ◇ The parameters `C` and `D` are the vertices of the edge `pos` of the graph G. `C` and `D` should be integers between 0 and $V-1$, inclusive. Also, `C` $\neq$ `D` should be satisfied. If `C` or `D` does not satisfy these conditions, your program is considered as **Wrong Answer [5]**.

Here, U and V are the integers specified by `InitG`.

In the function `Alice`, after calling the function `InitG` once, the function `MakeG` should be called exactly U times. If the function `InitG` is called twice, your program is considered as **Wrong Answer [6]**. If the function `MakeG` is called before the function `InitG` is called, your program is considered as **Wrong Answer [7]**. If `InitG` is not called when the function `Alice` terminates, or, the function `MakeG` is not called U times, your program is considered as **Wrong Answer [8]**. When the function `Alice` terminates, if the graph G described by `Alice` is not a simple graph, your program is considered as **Wrong Answer [9]**.

If the call to the function `Alice` is considered Wrong Answer, your program is terminated immediately.

The second file is `Bob.cpp`. This file outputs, given information of the graph G received by Bob, the airline route map of JOI Kingdom. It should implement the following function. The program should include `Boblib.h`.

- `void Bob( int V, int U, int C[], int D[] )`

  For each test case, this function is called once.

  - The parameter `V` is the number of vertices of the graph G.
  - The parameter `U` is the number of edges of the graph G.
  - The parameters `C[]`, `D[]` are sequences of length U describing the edges of the graph G.

  Using the following functions, the function `Bob` recovers the airline route map of JOI Kingdom, and outputs information of the airline route map.

  ★ `void InitMap( int N, int M )`

  This function specifies the number of islands of JOI Kingdom, and the number of airline routes in JOI Kingdom.

  ◇ The parameter `N` is the recovered number of islands in JOI Kingdom. `N` is an integer which should be equal to the actual number of islands in JOI Kingdom. If they are not equal, your program is considered as **Wrong Answer [10]**.

  ◇ The parameter `M` is the recovered number of airline routes in JOI Kingdom. `M` is an integer which should be equal to the actual number of airline routes in JOI Kingdom. If they are not equal, your program is considered as **Wrong Answer [11]**.

  ★ `void MakeMap( int A, int B )`

  This function specifies the number of airline routes in JOI Kingdom.

  ◇ The parameters `A` and `B` mean there is an airline route connecting the island `A` and the island `B`. `A` and `B` are integers between 0 and $N - 1$, inclusive. Also, `A` $\neq$ `B` should be satisfied. If `A` or `B` does not satisfy these conditions, your program is considered as **Wrong Answer [12]**. If there does not exist an airline route connecting the island `A` and the island `B` in JOI Kingdom, your program is considered as **Wrong Answer [13]**. The airline route described by a call to this function should be different from the airline routes of previous calls. When `MakeMap( A, B )` is called, if either `MakeMap( A, B )` or `MakeMap( B, A )` was already called before, your program is considered as **Wrong Answer [14]**.

Here, N is the integer value specified by `InitMap`.

In the function `Bob`, after calling the function `InitMap` once, the function `MakeMap` should be called exactly M times. If the function `InitMap` is called twice, your program is considered as **Wrong Answer [15]**. If the function `MakeMap` is called before the function `InitMap` is called, your program is considered as **Wrong Answer [16]**. If `InitMap` is not called when the function `Bob` terminates, or, the function `MakeMap` is not called M times, your program is considered as **Wrong Answer [17]**. Here, M is the integer value specified by `InitMap`.

If the call to the function `Bob` is considered Wrong Answer, your program is terminated immediately.

Grading Procedure

The grading is done in the following way. If your program is considered as Wrong Answer, it is terminated immediately.

(1) The function `Alice` is called once whose parameters describe information of the airline route map of JOI Kingdom.

(2) Let G be the graph specified by the function `Alice`. The function `Bob` is called once whose parameters are the shuffled numbers of the vertices of G and the shuffled numbers of the edges of G.

(3) Your program is graded.

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 `static` to avoid confliction with other files. When it is graded, it will be executed as two processes of Alice and Bob. The process of Alice and the process of Bob can not share global variables.

- Your program should not use the standard input and the standard output. Your program should not communicate with other files by any methods. But, 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 consists of one source file, which is `grader.cpp`. If your programs are `Alice.cpp` and `Bob.cpp`, to test them, your put these files (`grader.cpp`, `Alice.cpp`, `Bob.cpp`), `Alicelib.h`, and `Boblib.h` in the same directory, and run the following commands to compile your programs.

```
g++ -std=c++14 -O2 -o grader grader.cpp Alice.cpp Bob.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.

- The first line contains two space separated integers This means JOI Kingdom consists of N islands, and there are M airline routes in JOI Kingdom.
- The following M lines contain information of the airline route map. The $(i + 1)$-th line $(0 \le i \le M - 1)$ of the M lines contains two space separated integers A_i, B_i. They describe information of the airline route map of JOI Kingdom.

Output of the Sample Grader

When the program terminates successfully, the sample grader writes the following information to the standard output. (The quotation mark is not written actually.)

- If your program is considered as Wrong Answer, the sample grader writes its type in the following form "`Wrong Answer [1]`" and terminates.
- If either of the calls to `Alice` and `Bob` are not considered as Wrong Answer, the sample grader writes "`Accepted`." It also outputs the value of V.

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

Constraints

All input data satisfy the following conditions.

- $1 \leq N \leq 1\,000$.
- $0 \leq M \leq N(N-1)/2$.
- $0 \leq A_i \leq N-1$ $(0 \leq i \leq M-1)$.
- $0 \leq B_i \leq N-1$ $(0 \leq i \leq M-1)$.
- $A_i \neq B_i$ $(0 \leq i \leq M-1)$.
- $(A_i, B_i) \neq (A_j, B_j)$ and $(A_i, B_i) \neq (B_j, A_j)$ $(0 \leq i < j \leq M-1)$.

Subtask

There are 3 subtasks. The score and additional constraints of each subtask are as follows:

Subtask 1 [22 points]

- $N \leq 10$.

Subtask 2 [15 points]

- $N \leq 40$.

Subtask 3 [63 points]

There are no additional constraints.

Grading

- In Subtask 1 or Subtask 2, if your program solves all of the test cases, you get full score.
- In Subtask 3, if your program solves all of the test cases, your score is calculated as follows. Let MaxDiff be the maximum of the difference $V - N$.
  - When $101 \leq \text{MaxDiff}$, your score is 0.
  - When $21 \leq \text{MaxDiff} \leq 100$, your score is $13 + \left[\dfrac{100 - \text{MaxDiff}}{4}\right]$. Here, $[x]$ is the largest integer not exceeding x.
  - When $13 \leq \text{MaxDiff} \leq 20$, your score is $33 + (20 - \text{MaxDiff}) \times 3$.
  - When $\text{MaxDiff} \leq 12$, your score is 63.

Sample Communication

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

Sample Input 1	Sample Calls			
	Call	Return	Call	Return
`4 3`	`Alice(...)`			
`0 1`			`InitG(4,3)`	
`0 2`				(none)
`0 3`			`MakeG(0,0,1)`	
				(none)
			`MakeG(1,0,2)`	
				(none)
			`MakeG(2,0,3)`	
				(none)
		(none)		
	`Bob(...)`			
			`InitMap(4,3)`	
				(none)
			`MakeMap(0,1)`	
				(none)
			`MakeMap(0,2)`	
				(none)
			`MakeMap(0,3)`	
				(none)
		(none)		

In this case, the parameters given to the functions `Alice(...)`, `Bob(...)` are as follows.

Parameters	`Alice(...)`	`Bob(...)`
`N`	4	
`M`	3	
`V`		4
`U`		3
`A`	$\{0, 0, 0\}$	
`B`	$\{1, 2, 3\}$	
`C`		$\{2, 2, 2\}$
`D`		$\{3, 0, 1\}$

Sample Input 2

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