Task 1: `Hello`

Mr. Panda wants to welcome all the contestants taking part in this year's NOI but he is too lazy to welcome each contestant one by one. Hence, he wants to write a script that automatically generates a greeting given a contestant's name. However, he is too busy setting questions for the NOI. Can you help him write this script?

Input format

Your program must read from standard input.

The input contains a single username consisting of English alphabets. The username contains no other characters including spaces, and contains at most 10 letters.

Output format

Your program must print to standard output.

Your program should print a welcoming message "Hello [username]!" where the username should be replaced with the username from the input.

Sample Testcase 1

Input	Output
`Panda`	`Hello Panda!`

Sample Testcase 2

Input	Output
`Kitty`	`Hello Kitty!`

Task 2: `Sum`

In the year 3018, NOI has grown so big that a total of N schools are sending students to participate for the NOI. The i-th school is sending X_i students to join the secondary division and Y_i students to join the junior college division. For administrative purposes, Mr. Panda needs your help the total number of students participating in each division.

Input format

Your program must read from standard input.

The input starts with a single integer, N, in a single line. N denotes the total number of schools. N lines will then follow with 2 integers each, the i-th line will contain X_i and Y_i. This indicates the i-th school sent X_i students for the secondary division and Y_i students for the junior college division.

Output format

Your program must print to standard output.

Your program should print two integers on a line separated by a space. The first integer is the total number of students in the secondary division and the second integer is the total number of students in the junior college division.

Subtasks

The maximum execution time on each instance is 1.0s.

For all test cases, $0 \leq X_i, Y_i \leq 10^9$ (Yes the population has had a massive increase!)

Your program will be tested on sets of input instances that satisfy the following restrictions:

Subtask	Marks	N
1	23	$2 \leq N \leq 2\,000$
2	33	$2 \leq N \leq 200\,000$
3	44	$2 \leq N \leq 10\,000\,000$

Sample Testcase 1

Input	Output
2 1 4 2 3	3 7

Task 3: `Sorting`

Mr. Panda has managed to infiltrate into the NUS servers and plans to get information about the N questions in the upcoming NOI. Unfortunately, Mr. Panda's skills are not good enough to get the questions themselves. However, he is able to get information about the difficulty of each question. He is also able to change the order the questions will come out for the actual NOI. Currently, the questions are not sorted in order of difficulty. To make his life easier, he would like to sort the questions in increasing order of difficulty from easiest to hardest.

Implementation Details

In this task, you are asked to implement the following function: (It is different from the normal task, which requires the program to read from standard input and output to standard output.)

```
void sort_questions(int N, int Q, int A[])
```

This function will be called exactly once with two input parameters and one output parameter. The input parameters are N and Q, the number of questions in NOI and the maximum number of times you can call the function below. The output parameter is A. You are required to determine the order of difficulty of the questions and return them in array A where $A[0]$ is the easiest question, $A[1]$ is the second easiest question and so on until $A[N-1]$ is the hardest question. The questions are labeled from 1 to N so A should contain each number from 1 to N exactly once.

You are allowed to call the following grader function to complete the task:

```
bool compare_difficulty(int X, int Y)
```

This function will return true if task X is easier than task Y and false otherwise. No two tasks will be of the same difficulty. If you call this function more than Q times or X or Y is not a valid problem number, the program will terminate immediately and you will be given a *Wrong Answer* verdict.

Consider an NOI with 3 problems where problem, 1, 2, 3 are rated with difficulty 3, 1, 6 respectively. A possible interaction between your program *sort_questions* and the grader function *compare_difficulty* could be as follows:

- `compare_difficulty(1,2)=false`

  Problem 1 is harder than problem 2

- `compare_difficulty(2,3)=true`

  Problem 2 is easier than problem 3

- `compare_difficulty(1,3)=true`

  Problem 1 is easier than problem 3

At this point, the contestant's *sort_questions* function decides it has enough information to deduce the order of the difficulty of the questions. It sets $A = [2, 1, 3]$ and then terminates. This would yield a correct answer and the contestant would pass that test case.

Subtasks

The maximum execution time on each instance is 1.0s.

Your program will be tested on sets of input instances that satisfy the following restrictions:

Subtask	Marks	Q	N
1	23	1	$N = 2$
2	33	500000	$2 \leq N \leq 1000$
3	44	12000	$2 \leq N \leq 1000$

Testing

You may download the grader file, *grader.cpp*, the header file, *sorting.h*, a solution template, *sorting.cpp*, and the compilation command, *compile.sh* under *Attachments*. Edit the solution template *sorting.cpp* directly when implementing your solution, and run *compile.sh* to compile the executable. A small and large test case for each subtask are also provided for your reference. The grader input format is provided below.

Grader Input Format

- one line with a two integers N and Q, the number of questions and the maximum number of times the function can be called
- N lines with one signed 32-bit integer each where the integer on the i-th line represents the difficulty of the i-th problem