Task	TUNNEL	TOWER	ASK

Source file	`tunnel.pas`
	`tunnel.c`
	`tunnel.cpp`
		`tower.pas`
		`tower.c`
		`tower.cpp`
			`ask.pas`
			`ask.c`
			`ask.cpp`

Input file	`tunnel.in`	`tower.in`	–
Output file	`tunnel.out`	`tower.out`	–
Time limit per test	10 seconds	10 seconds	10 seconds
Number of tests	10	10	10
Points per test	3	5	7
Total points	**30**	**50**	**70**
	150		

TUNNEL

Mirko stands at the **entrance** and Slavko stands at the **exit** of the tunnel. They **note** the registration plates of cars that pass by them and **provide** that information to police patrol that is a few miles down the road.

Using the information that Mirko and Slavko gave them, police can determine, without making mistake, that some drivers made an overtake while driving through the tunnel, which is strictly forbidden.

Write a program that will determine number of drivers for which the police can **certainly claim** that they made an overtake.

We assume that traffic through the tunnel was without any stops.

Input data

Input file consist of 2N+1 lines.

In first line there is an integer N, $1 \leq N \leq 1000$, number of cars.

In next N lines there are registration plates of those cars, in order they **entered** the tunnel.

In next N lines there are registration plates of those cars, in order they **exit** the tunnel.

Registration plate of some car consists of at least **six** and at most **eight** characters and only allowed characters are capital letters of English alphabet (A-Z) and digits of decimal number system (0-9).

Output data

First and only line of output file should be the number of drivers who police can certainly punish for making overtake in the tunnel.

Examples

tunnel.in

```
4
ZG431SN
ZG5080K
ST123D
ZG206A
ZG206A
ZG431SN
ZG5080K
ST123D
```

tunnel.out

```
1
```

tunnel.in

```
5
ZG508OK
PU305A
RI604B
ZG206A
ZG232ZF
PU305A
ZG232ZF
ZG206A
ZG508OK
RI604B
```

tunnel.out

```
3
```

tunnel.in

```
5
ZG206A
PU234Q
OS945CK
ZG431SN
ZG5962J
ZG5962J
OS945CK
ZG206A
PU234Q
ZG431SN
```

tunnel.out

```
2
```

TOWER

There are empty plastic glasses marked with numbers from 1 to N which can be put one into another, standing on the table.

You are given number of moves **a→b** that mean that you should move the bunch containing glass **a** to the bunch containing glass **b**. If **a** is equal to **b**, or glasses **a** and **b** are in the same bunch, than that move does nothing.

For example, we have 7 glasses and moves 1→3, 2→6, 3→6, 4→7 i 4→2, then situation on the table will change as follows:

```
                                                                      4
                                                                      7
                                                    1        1        1
                                                    3        3        3
                    1            1     2            2        2 4      2
1 2 3 4 5 6 7 --> 2 3 4 5 6 7 --> 3 4 5 6 7 --> 4 5 6 7 --> 5 6 7 --> 5 6
```

Moves must be made in the given order.

Write a program that will choose **zero** move, which is move that should be made before the first move in the given order. You should choose such zero move that will **maximize** the number of glasses in the **largest** bunch.

Input data

In the first line of input there are two integers N and M, $2 \leq N \leq 10000$, $0 \leq M \leq 100000$. N is the number of glasses standing at the table and M is the number of moves to be made.

Next M lines contain moves to be made. In each of those lines are two integers **a** and **b** which represent move **a→b**.

Output data

One and only line of output file should contain zero move **a→b** represented by two integers **a** and **b** separated by one space.

Note: solution need not to be unique.

Examples

tower.in

```
5 3
1 2
5 3
4 1
```

tower.out

```
4 5
```

tower.in

```
6 4
5 2
3 3
1 3
6 2
```

tower.out

```
3 5
```

tower.in

```
10 9
3 7
5 9
10 10
9 5
8 7
3 7
6 1
4 2
2 6
```

tower.out

```
7 2
```

Write a program that guesses imagined string that consists **only** of characters 0 and 1 and its length is **unknown**.

Maximal length of imagined string will be **100** characters, and your program must guess it in **at most 300 calls** to function `Ask`.

Library

For solving this task you are provided with library `lib` containing three functions:

`Init` – you need to call it exactly once at the beginning of your program with no arguments. This function doesn't return and value.

```
procedure Init;
void Init(void);
```

`Ask` – function that is called with string of characters 0 and 1 as argument. This function returns value **1** if parameter string is substring of unknown string (not necessarily consecutive) and **0** otherwise. You are allowed to call this function at most 300 times.

```
function Ask(a : string) : integer;
int Ask(const char *a);
```

`Solution` – you should call this function at the end of your program; as an argument you should pass string you claim is identical to the imagined string. This function does not return any value and it regularly ends execution of your program.

```
procedure Solution(a : string);
void Solution(const char *a);
```

Library `lib` generates file `ask.log`. It consists of all questions that your program asked with given answer. In case there are more then 300 questions, in last, 301st row of this file there will be written message '`Number of questions is more then 300!`' If you call function Solution after at most 300 questions in last line there will be written message: '`Correct answer after ??? questions.`' or 'Wrong answer.'

Instructions for Pascal programmers

At the beginning of your program write '`uses lib`'.

Also, you need to copy files `lib.o` i `lib.ppu` from directory `c:\dmih\pita\lib_pas` to your current directory.

Instruction for C programmers

You need to copy files `lib.h` i `lib.o` from directory `c:\dmih\pita\lib_c` in you current directory.

In RHIDE open project (`Project->Open`) and call him `ask`. Using `Project->AddItem` add files `ask.c` (your program) and `lib.o`.

At the beginning of program write '`#include "lib.h"`'.

Important: don't use option `Compile->BuildAll` (because it erases file `lib.o`).

Testing

You can test your program by creating file `ask.in` in which you will write some string of 0 and 1. That string will be imagined string used by library.

Notice: for evaluation we will use library without that test option.

Example

`ask.in`

```
1000101
```

String of 5 correct calls to functions from library would be, for example:

`Pascal`

```
Init;
Ask('101');
Ask('11');
Ask('10001001');
Solution('1000101');
```

`C`

```
Init();
Ask("101");
Ask("11");
Ask("10001001");
Solution("1000101");
```

In file `ask.log` will then be:

```
Question 1 : 101 ..... YES
Question 2 : 11 ..... NO
Question 3 : 10001001 ..... NO
Solution is 1000101 ..... YES
Correct answer after 3 questions.
```