UNDERGROUND CITY

PROBLEM

You are imprisoned in one of the underground cities of Cappadocia. Wandering around in the dark you find by chance the map of the city. Unfortunately, there is no mark on the map pointing where you are. It is your task to find that out by exploring the city.

The map of the city is a rectangular grid of unit squares. Each square is either an open square, marked with the letter '`O`', or a part of a wall, marked with the letter '`W`'. The north direction is also shown on the map. Luckily, you have a compass at hand so you can orient the map correctly. Initially, you are on an open square.

Everything starts by calling the procedure (or function) `start` with no arguments. You can explore the city by using the procedures (or functions) `look` and `move`.

You can pose questions in the form of a function call `look(`*dir*`)` where *dir* denotes the direction you are looking at, which can be one of the characters '`N`', '`S`', '`E`' and '`W`' denoting north, south, east and west, respectively. Now assume the argument *dir* is '`N`'. The reply will be the letter '`O`' if the square to your north is an open square, and '`W`' if it is a wall. Similarly, it is possible to look at and get information about the other neighboring squares.

You can step into one of the four neighboring squares by calling `move(`*dir*`)` where *dir* denotes the direction of your step as described above. You can only move to an open square. Attempting to move into a wall would be a grave mistake. It is possible to reach any open square of the city starting from any open square.

You are required to find the position of the open square where you found the map by looking (calling `look(`*dir*`)`) minimum number of times. Once you found the position you must report it by calling `finish(`*x*`,`*y*`)` where *x* is the horizontal (west-east) coordinate and *y* is the vertical (south-north) coordinate of the position.

ASSUMPTIONS

- $3 \le U \le 100$ where U is the width of the map, i.e. length, in number of squares, of the map in the horizontal (west-east) direction.
- $3 \le V \le 100$ where V is the height of the map, i.e. length, in number of squares, of the map in the vertical (south-north) direction.
- The city is surrounded with walls, which are included on the map.
- The south-west corner of the city has the coordinate (1,1) and the north-east corner has the coordinate (U,V).

INPUT

The input is a text file named **`under.inp`**.

- The first line contains two numbers: *U*, *V*.
- Each of the following *V* lines contains a row of the map in the horizontal direction. Each line consists of *U* characters, so that the *x*'th character on the (*V*-*y*+2)'th line of the input file has information about the position (*x*,*y*) of the map: It is either a letter '`W`' denoting a wall, or a letter '`O`' denoting an open square. The data on these lines do not have any blanks in between.

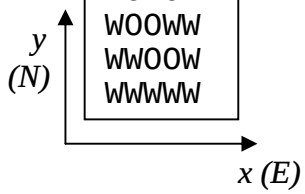
OUTPUT

No output file will be generated. The result found by your program must be reported by calling `finish(`*x*,*y*`)`.

EXAMPLE

`under.inp`:

```
5 8
WWWWW
WWWOW
WWWOW
WOOOW
WOWOW
WOOWW
WWOOW
WWWWW
```



A possible interaction which ends with the correct `finish` call:

Interaction:	
start()	
look('N')	'W'
look('E')	'O'
move('E')	
look('E')	'W'
finish(3,5)	

INSTRUCTIONS FOR PASCAL PROGRAMMERS

Have the following in your source file:

```
uses  undertpu;
```

This tpu will provide the following:

```
procedure start; { must be called first }
function look (dir:char):char;
procedure move (dir:char);
procedure finish (x,y:integer); { must be called last }
```

INSTRUCTIONS FOR C/C++ PROGRAMMERS

Have the following in your source file:

```
#include "under.h"
```

This will provide the following declarations:

```
void start (void); /* must be called first */
char look (char);
void move (char);
void finish (int,int); /* must be called last */
```

Also create a project called `under` which should include your program and the library for interaction named `underobj.obj`. To do this you need to use the *project*

menu of IDE and choose the *open* option to create a project, and then use *add item* to include your source file (`under.c` or `under.cpp`) and the file `underobj.obj`. Use the LARGE memory model compiler option. (*Careful*: This overrides the memory model mentioned in the *Rules of Contest*.)

EVALUATION

Your program will be allowed to run 5 seconds.

To get full credit, *A*, for a test case the number of calls to `look`, *x*, must be less than or equal to the number *M*, set by the evaluation program. Note that *M* is chosen as larger than (>) the minimum. In particular, *M* is independent of the clockwise or counter-clockwise ordering of the directions for looking. You can obtain partial credit if the number of calls to `look` is greater than (>) *M* but less than (<) twice *M*. The points you get is calculated by rounding to the nearest integer the value obtained by the following formula:

A	if $x \leq M$
$A\,(2M - x)\,/M$	if $M < x < 2M$
0	if $x \geq 2M$

Illegal behavior by your program will result in zero points. Illegal behaviors specific to this task are the following:

- Calling a library procedure (or function) with an unacceptable argument, for example a character which does not designate a direction.
- Attempting to move into a wall.
- Failing to follow the instructions.

HOW TO TRY OUT YOUR PROGRAM

Create a text file called `place.txt` including the position of the map. Run your program. See the result in the file `result.txt`.

The file `place.txt` should have one line having the horizontal and vertical coordinates of the position of the map. You need to create your own input data file `under.inp`. The `result.txt` file will contain two lines. The first line will have the arguments *x* and *y* for your call to `finish` (*x*,*y*). The second line will have a message of the form "`You used look nnn times`". Note that this trial is for checking the compatibility of your program with the library. It has nothing to do with the correctness of your solution.