South African Computer Olympiad
Final Round
Day 2

Overview

Author	**Migael Strydom**	**Charles Bradshaw and Ben Steenhuisen**	**Keegan Carruthers-Smith**
Problem	**factor**	**taunter**	**parrots**
Source	factor.java factor.py factor.c factor.cpp factor.pas	taunter.java taunter.py taunter.c taunter.cpp taunter.pas	parrots.java parrots.py parrots.c parrots.cpp parrots.pas
Input file	factor.in	taunter.in	stdin
Output file	factor.out	taunter.out	stdout
Time limit	1 second	1 second	1 second
Number of tests	10	10	10
Points per test	10	10	10
Total points	**100**	**100**	**100**

The maximum total score is 300 points.

Factorise

Author

Migael Strydom

Introduction

You are at the argument clinic having an argument about prime numbers. The dispute is about a particular number, which you are told is a prime number. You are not so sure, however, and want to write a program that factorises numbers into their prime factors so that you can win this debate.

Task

You will be given an integer N that needs to be factorised into its prime factors. Recall that a *prime number* is a number that is divisible only by 1 and itself, and a *factor* of N is a number that perfectly divides into N. 1 is not a prime number.

In some cases a certain prime factor divides into N more than once. Your program will also have to output the number of times each prime factor divides into N.

If the given integer N is prime, then it has only one prime factor, which is N itself.

Example

Take, for example, the number 12. It can be written as $12 = 2 \times 2 \times 3$. 2 and 3 are both prime numbers. Note that the prime factor 2 appears twice, and the prime factor 3 appears once.

Therefore, when given the number 12, the correct answer is that the prime factors are 2, which appears twice, and 3, which appears once.

Input (factor.in)

The input is a single line containing an integer, N.

Sample input

```
12
```

Output (factor.out)

The output should consist of P lines, where P is the number of distinct prime factors of N. Each line should contain two space-separated integers, f_i and m_i, where f_i is a prime factor, and m_i is the number of times f_i divides N.

The output should be sorted in ascending order by the f_i.

Sample output

```
2 2
3 1
```

Constraints

$2 \leq N \leq 100000$

Time limit

1 second.

Scoring

A correct solution will score 100%, while an incorrect solution will score 0%.

French Taunter

Author

Charles Bradshaw and Ben Steenhuisen

Introduction

The French Taunter has too many taunts to use, and has therefore decided (using his vast deductive capabilities) to hide all his favourite words in a rectangular grid of seemingly random letters. To distinguish between the nouns and the adjectives, the Taunter put all nouns horizontally in the grid, and all adjectives vertically. Due to the versatility in the Taunter's interpretation of the English language, some of his words are both adjectives and nouns, and hence are stored both vertically and horizontally. The cunning Taunter decided that some words are especially effective, and so wrote them backwards in the grid.

However, the Taunter has since developed a new list of words he wants to use (he has developed the list over a long period of time, and so there may be words that appear multiple times on the list). Before he adds them to the array, he needs to find out if each word has been inserted already. Never one to admit to the foolishness of his own data structures, he has asked you to search for words in his list that are already in the grid.

Task

Given an $N \times M$ array of upper-case letters and W upper-case words, you must detect which of the W words are in the grid horizontally (forwards and/or backwards), vertically (up and/or down), both (horizontally and vertically) or neither.

Example

Given the grid in Figure 1 and the list of words to search for in Table 1:

T	H	G	I	N	K
S	O	A	R	I	H
D	F	F	O	U	R
S	T	O	N	E	S

Figure 1: An example grid

NI
IRON
KNIGHT
HAMSTER
STONES

Table 1: The list of words to search for in Figure 1

- `NI` appears in the 5th column downwards, and in the first row backwards, so it is both horizontal and vertical.
- `IRON` appears in the 4th column downwards, so it is only vertical.
- `KNIGHT` appears in the first row backwards, so is only horizontal.
- `HAMSTER` does not appear, so it is neither.
- `STONES` appears in the last row, so it is only horizontal.

Input (taunter.in)

The first line contains three space-separated integers N, M and W. The next N lines each contain M upper-case letters which describe the grid. The next W lines each contain a single string of upper-case letters, the words to be searched for in the grid.

Sample input

```
4 6 5
THGINK
SOARIH
DFFOUR
STONES
NI
IRON
KNIGHT
HAMSTER
STONES
```

Output (taunter.out)

The output must be W lines describing the presence of the words in the grid. The i^{th} line should contain:

- `HORIZONTAL` if the i^{th} word appears only horizontally in the grid
- `VERTICAL` if it appears only vertically in the grid

- `BOTH` if it appears both horizontally and vertically in the grid, or
- `NEITHER` if the word does not appear at all in the grid

Sample output

```
BOTH
VERTICAL
HORIZONTAL
NEITHER
HORIZONTAL
```

Constraints

- $1 \leq N, M \leq 50$
- $1 \leq W \leq 40000$
- $\text{length}(\text{word}_i) \leq \min(N, M)$

Additionally, in 50% of the test-cases:

- $1 \leq N, M \leq 15$
- $1 \leq W \leq 10000$

Time limit

1 second.

Scoring

If you get the solution correct for all the words you will score 100%. If you output that a word could be found either vertically or horizontally in the grid (or both) and this is actually not possible, then you will score 0%. Otherwise, if you get the correct answer for K of the W words, you will score $50 \times \frac{K}{W}\%$ rounded down.

Living Dead Parrot

Author

Keegan Carruthers-Smith

Introduction

The Pet Shop Owner has a lot of cages with parrots in them at the back of his store. The owner needs to find out which cages have parrots that are alive, but the door to the back of his store is locked. The cages are lined up in the back and each cage is attached to a rope. Each rope goes through a pulley system to the front of the store. Pulling these ropes from the front of the store causes the cages to rattle in the back. A parrot will squawk if its cage is rattled.

The pulley system causes a range of cages that are adjacent to each other to rattle. The Pet Shop Owner has been using this pulley system for many years and has learned how to pull the ropes such that he can rattle any range of cages he wants.

Task

You must work out which cages have parrots that are alive in them. The owner can work out how many live parrots are in the range a to b by listening to the number of squawks. The owner's time is limited, so he can only pull on the pulley up to M times. The cages are numbered from 1 to N. There are K live parrots.

Input/Output

You must read input from standard input (as you would normally read input from the keyboard). You should begin by reading in a line containing the space-separated integers N and M, the number of cages and the number of pulls the shop owner can make respectively.

From here you have 2 possible responses, which you write to standard output (as you would normally write to the console):

- **Range Query:** Q a b
- **Answer:** A S

The range query will return the number of parrots that are alive in the range a to b, including a and b.

You must remember to flush the output stream after every query, to ensure that the evaluator receives it.

- Pascal: `flush(output);`
- C stdio: `fflush(stdout);`
- C++ iostreams: `cout << flush;`
- Java: `System.out.flush();`
- Python: `sys.stdout.flush()`
  `# Requires 'import sys'`

The answer, S, is a space-separated sequence of 0's and 1's representing whether the parrot in each cage is alive (1) or dead(0).

You can only make up to M range queries, and give one answer. After you have given an answer your program must terminate without producing further output.

Example

For the sake of clarity in this example, we shall ignore the constraints on the number of parrots that are alive, K, as well as the number of questions allowed, M.

See Table 2 for an example of the interaction with the evaluator when $N = 20$ and $M = 10$.

Input	Output
20 10	
	Q 5 10
0	
	Q 11 20
2	
	Q 17 19
2	
	Q 18 19
2	
	Q 1 5
1	
	Q 3 3
1	
	A 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0

Table 2: Sample interaction. $N = 20$, $M = 10$ and cages 3, 18 and 19 contain alive parrots.

Testing

During the contest, you may wish to test your program using some set of dead and living parrots. The format of the file that you should upload to the server to test your program with is as follows. The first line should contain a single integer N, the number of cages. The next line should contain N space-separated integers, each of which is either 0 or 1 to indicate whether the parrot is dead or alive respectively.

Please take note of the constraint below on K, the number of live parrots — if your test-case has no live parrots or too many live parrots, then the test will fail.

Constraints

- $100 \leq N \leq 100000 = 10^5$
- $M = \lceil \log_2(N) \times \frac{N}{100} \rceil$
- $1 \leq K \leq \frac{N}{100}$

Note that $\lceil x \rceil$ is the smallest integer greater than or equal to x.

Time limit

1 second.

Scoring

If you ask more than M range queries or output a parrot as alive when it is dead you will receive 0% for the test case. You will also receive 0% if you make any range query that does not satisfy $1 \leq a \leq b \leq N$, if any of your outputs are formatting incorrectly (e.g., outputting values other than 0's and 1's in S).

You will receive 100% for a test case if you guess the positions of all parrots correctly.

Otherwise, if you correctly identify the positions of D live parrots, you will get $60 \times \frac{D}{K}$%.