University of Arkansas
Department of Computer Science and Computer Engineering
2016 High School Programming Contest
Problems

Wing Ning Li   Hung Nguyen   Adam Higgins   Paul Martin   Tyler Moore

Revision Date: March 11, 2016

Please carefully read the output specification. Submitted solutions are graded by a differential program, i.e., the output produced by a submission will be compared *byte-for-byte* with the output of the judges' solutions. Thus, even though submitted solutions may produce output that is *technically* correct, it will be counted as incorrect if these specifications are not followed exactly.

For all problems, solutions receive input from standard input (*stdin*) and produce output on standard output (*stdout*).

Restricting input and output to stdin and stdout does not prevent a program from processing files. Let `Main` be the program name, `Input` be the input file name, and `Output` be the output file name. The following command at the terminal allows the program to read and write the input and output files respectively:

```
./Main < Input > Output      # for C or C++, assuming Main is the executable
java Main < Input > Output   # for Java, assuming Main is a class file
```

For all problems, an *integer* is defined as an optional minus sign followed by one digit from the set 1..9 followed by zero or more digits from the set 0..9. `3`, `-21`, and `309` are integers; `0123`, `- 4`, and `+38` are not.

A *positive integer* is an integer whose value is greater than 0. You may assume the value of a positive integer is no more than $2^{31} - 1$, or $2,147,483,647$.

A *string*'s maximum length is 16,384.

For all problems, a *line* is defined as zero or more characters followed by a newline ('\n') character. Unless otherwise directed, a line of output should have no leading or trailing whitespace. A program producing the integer 27 on one line of output should consist of the character '2' followed by the character '7' followed by the newline ('\n') character.

Input for a problem consists of multiple test cases; each test case is followed by a blank line. You must read input until the *end-of-file*. Refer to the code for the Warm Up problem for hints and ideas for handling input properly.

For each test case, your program must output the result following the specified format in each problem. Output a blank line after the output of each test case. Refer to the code for the Warm Up problem again for hints and ideas for handling output properly.

1 Does the integer have 2 as its factor?

An integer x has 2 as a factor if $x = 2 \times k$ for some integer k. For example, $2 = 2 \times 1$ (here $k = 1$) and $4 = 2 \times 2$ (here $k = 2$) have 2 as a factor; and 3 and 5 do not have 2 as a factor.

Given an integer n, decide if n has 2 as its factor or not.

input Each test case has only one line containing an integer n where $1 \leq n \leq 2^{31} - 1$. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, output two lines: the first line containing either `YES` or `NO`, where `YES` means the input integer has 2 as its factor, and `NO` means otherwise; *and a blank line as second*

Example:

Input

```
2

3

4

5

```

Output

```
YES

NO

YES

NO
```

2 What is the factor that occurs the most often?

To generalize the notion of a factor, an integer x has f as a factor if $x = f \times k$ for some integer k where $k \neq 1$ and $k \neq x$ (we exclude both 1 and the number itself as its factor). For example, $12 = 6 \times 2$ (here $k = 2$), $12 = 4 \times 3$ (here $k = 3$), $12 = 3 \times 4$ (here $k = 4$), and $12 = 2 \times 6$ (here $k = 6$), so 6, 4, 3, 2 are factors of 12. From the definition, 2 and 3 does not have any factor. For them the factor that occurs the most often is NONE. On the other hand, $12 = 2 \times 2 \times 3$. In $2 \times 2 \times 3$, 2 occurs two times and in all other product expressions each factor appears only once. Hence for 12, the factor that occurs the most often is 2.

Given an integer n, decide the factor that occurs the most often for n. In the case of a tie, the smallest factor is the answer.

input Each test case has only one line containing an integer n where $1 \leq n \leq 2^{31} - 1$. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, output two lines: the first line containing the factor that occurs the most often for the input; *and a blank line as second*

Example:

Input

```
2

3

12

36

108
```

Output

```
NONE

NONE

2

2

3
```

3 Cell Adjacency

A grid board consists of square cells. Two cells are called adjacent if they share a side (and two corners) or a corner. In another way, if a cell is not on the edge of the board, it should have 8 adjacent cells. A cell can be either occupied or empty.

Write a program to determine the maximum number of occupied cells that are adjacent to at least another occupied cell.

input Each test case contains 2 sections. The first line has $h (1 \le h \le 10000)$, the height, and $w (1 \le w \le 10000)$, the width, of the grid. The two values are separated by a space. The next h lines will contain w cells with *x*, an occupied cell, or ., an empty cell. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, print the number of occupied cells that are adjacent to at least one other occupied cell *and a blank line.*

Example:

Input

```
2 3
x..
x.x

3 3
x..
.x.
...

```

Output

```
2

2
```

4 Gomoku

Please refer to the previous problem for the description of cell adjacency. Gomoku is a game where players alternate placing tokens on a grid attempting to get 5 tokens in a ”row”, where all the tokens must be either vertically aligned to one another, or horizontally aligned to one another, or diagonally aligned to one another.

Write a program to determine if a Gomoku game board has a winner. You may assume that any game board cannot have both players as winners.

input Each test case contains 15 lines. The 15 lines contain 15 characters: *W*, a white token; *B*, a black token; or ., an empty space. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, one line containing ”Black” if black has 5 in a row, ”White” if white has 5 in a row, or ”None” if neither player has won *and a blank line.*

Example:

Input

```
...............
...............
...............
...............
...............
..BBBBB........
...............
...............
...............
...............
...............
........WWWW...
...............
...............
...............

```

Output

```
Black
```

5 No Alias

An alias is another name for some person who already has a name. Names are encoded as integers. Hence two integers are aliases of one another if the corresponding names are aliases. Two aliases are represented as a pair of integers.

For example, for integers 1 to 10, we have the following aliases (1 8), (8 3), (2 7). The values that do not have alias are 4, 5, 6, 9, and 10.

Given an positive integer n and a set of alias pairs, decide those values from 1 to n that do not have any alias. You may assume that an integer cannot be an alias to itself.

input Each test case has several lines. The first line contains a positive integer n where $1 \leq n \leq 10,000$. The rest of the lines are the same as the number of pairs and each line contains two integer separated by a space. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, output two lines: the first line containing those integers, in increasing order separated by a space, that do not have any alias or NONE if all have at least an alias *and a blank line as second*

Example:

Input

```
10
1 8
8 3
2 7

8
1 8
8 3
2 7

```

Output

```
4 5 6 9 10

4 5 6
```

6 All Aliases

The previous problem identifies all the integers that do not have any alias. In this problem, we list integers by groups where all integers that are aliases of the same person belong to the same group

input Each test case has several lines. The first line contains a positive integer n where $1 \leq n \leq 10,000$. The rest of the lines are the same as the number of pairs and each line contains two integer separated by a space. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, output several lines: each line containing those integers, in increasing order separated by a space, that are aliases of each other. The first element of each line are sorted increasingly. Output NONE if no alias exists. *There is a blank line to end the output.*

Example:

Input

```
10
1 8
8 3
2 7

8
1 8
8 3
2 7
7 8

8

```

Output

```
1 3 8
2 7

1 2 3 7 8

NONE
```

7 Run-length Encoding

A run in a text string is a single alphabetic character (an alphabet character is an English letter, which could be lower case or upper case) that occurs some number of times consecutively. When a long string has many runs, it may become more space efficient to store a run-length encoding of the string rather than the full string itself. A run-length encoding of a string consists of an integer (not in the original string) followed by the first character of the original string, followed by another integer and the next character of the original string not identical to the first character, and so on. Each integer describes the length of the run of the associated character.

Given a text string, produce a run-length encoding of that text string.

input Each test case has only one line containing a text string, where the length of the string is between 1 and 10000. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, output two lines: the first line containing a run length encoding of the input; *and a blank line as second*

Example:

Input

```
AAAA

AAAABBB

ABBCCC

```

Output

```
4A

4A3B

1A2B3C
```

8 Valid Run-length Encodings

Please refer to the previous problem for the description of run-length encoding.

Given a text string and a prospective run-length encoding, decide if said run-length encoding is valid. If it is not, decode the prospective run-length encoding and output its corresponding text string.

input Each test case has two lines: the first line containing a text string, where the length of the string is between 1 and 10000, and the second line containing a prospective run-length encoding. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, output two lines: the first line containing either the output VALID in the case that the prospective run-length encoding is for the given text string, or a text string for which the prospective run-length encoding decodes; *and a blank line as second*

Example:

Input

```
AAAA
4A

AAAABBB
3A4B

ABBCCC
1A2B3C1D

```

Output

```
VALID

AAABBBB

ABBCCCD
```