Old Mutual - CSSA Computer Olympiad DAY 2

Problems Overview

Problem	Factorial Factors	Stone Game	Parity
Program name	FACT.EXE	STONE.EXE	PARITY.EXE
	FACT.PAS	STONE.PAS	PARITY.PAS
Source name	FACT.JAV	STONE.JAV	PARITY.JAV
	FACT.CPP	STONE.CPP	PARITY.CPP
Input file	FACT.IN	STONE.IN	PARITY.IN
Output file	FACT.OUT	STONE.OUT	PARITY.OUT
Time limit per test	20 seconds	20 seconds	20 seconds
Number of tests	10	10	10
Points per test	3	3	4
Total points	**30**	**30**	**40**

The maximum total score for Round 2 is 100 points.

Directory structure:

```
DAY1
 ________________
                 |FACT
                 |STONE
                 |PARITY
```

Factorial Factors (30)

Description

The factorial of a number N (written N!) is defined as the product of all the integers from 1 to N.
I.e. 5! = 5x4x3x2x1 = 120.

It is not surprising that factorials grow very rapidly, for instance 5! = 120 whereas 10! = 3 628 800.

One way of describing such large numbers is by specifying the number of times each prime number occurs in it.

If we consider 825 we could specify it as (0 1 2 0 1) meaning: no twos, 1 three, 2 fives, no sevens and 1 eleven. In other words 3 x 5 x 5 x 11 = 825.

Task

Write a program that will read in a number N and write out its factorial in terms of the numbers of the primes it contains.

Input:

The input will consist of a single `number`, N, ($2 \leq N \leq 100$).

Output:

The output will be a list of the number of times each prime occurs in N! (starting at the smallest prime).

Note: A prime number is not divisible by any number other than 1 and itself, (e.g. 2, 3, 5, 7, 11, 13...)

Example 1:

Input

```
5
```

Output

```
3 1 1
```

i.e. 5! Contains 3 two's, 1 three and 1 five

Example 2:

Input

```
53
```

Output

```
49 23 12 8 4 4 3 2 2 1 1 1 1 1 1 1
```

Stone Game (30)

Description

On the chessboard of size 4x4 there are 8 white and 8 black stones, i.e. one stone on each field. Such a configuration of stones is called a game position. Two stones are adjacent if they are on squares with a common side (i.e. they are adjacent in either horizontal or vertical direction but not diagonal). It means that each stone has at most four 4 neighbours. The only legal move in our game is exchanging any two adjacent stones.

Task

You are required to find the shortest sequence of moves transforming a given starting game position into a given final game position.

Starting Game Position

1	1	1	1
0	0	0	0
1	1	1	0
0	1	0	1

1	0	1	0
0	1	0	1
0	0	1	0
1	1	0	1

Final Game Position

Input

The starting game position is described in the first 4 lines of input file GAME.IN. There are 4 symbols in each line, which define the colour of each stone in the row from the left to the right. The lines describe the rows of the chessboard from the top to the bottom. Symbol `0' means a white stone and symbol `1' a black one. There is no space symbol separating the symbols in the line. The fifth line is empty. The next 4 following lines describe the final game position in the same way.

Output

The first line of output file GAME.OUT contains the number of the moves. The following lines describe the sequence of moves during the game. One line describes one move and it contains 4 positive integers R_1 C_1 R_2 C_2 separated by single spaces. These are the coordinates of the adjacent fields for the move, i.e. fields [R_1, C_1] and [R_2, C_2], where R_1 (or R_2) is the number of the row and C_1 (or C_2) is the number of the column. The rows on the chessboard are numbered from 1 (top row) to 4 (bottom row) and the columns are numbered from 1 (the leftmost column) to 4 (the rightmost one) as well (i.e. the coordinates of the left upper field are [1,1]). If there are multiple shortest sequences of moves transforming the starting position to the final position, you can output any one of them.

Example 1

Input

```
1111
0000
1111
0000

1110
0001
0111
1000
```

Output

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

Example 2

Input

```
1111
0000
1110
0010

1010
0101
0010
1101
```

Output (one of the correct solutions)

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

Parity (40)

Description

Imagine you play the following game with your friend. Your friend writes down a sequence consisting of zeroes and ones. You choose a continuous subsequence (for example the subsequence from the third to the fifth digit inclusively) and ask him, whether this subsequence contains even or odd number of ones. Your friend answers your question and you can ask him about another subsequence and so on. The object of the game is to guess the entire sequence of numbers.

Task

You suspect some of your friend's answers may not be correct and you want to convict him of falsehood and you decide to write a program to do this.

The program will receive a series of your questions together with the answers you have received from your friend. The aim of this program is to find the first answer that is probably wrong. I.e. that there exists a sequence that satisfies the answers to all the previous questions, but no such sequence satisfies this answer.

Input

The first line of input file PARITY.IN contains one number, which is the length of the sequence of zeroes and ones. This length is less or equal to 1000000000. In the second line, there is one positive integer, which is the number of questions asked and answers to them. The number of questions and answers is less or equal to 5 000. The remaining lines specify questions and answers. Each line contains one question and the answer to this question: two integers (the position of the first and last digit in the chosen subsequence) and one word which is either 'even' or 'odd' (the answer, i.e. the parity of the number of ones in the chosen subsequence, where 'even' means an even number of ones and 'odd' means an odd number).

Output

There is only one line in output file PARITY.OUT containing one integer X. Number X says that there exists a sequence of zeroes and ones satisfying first X parity conditions, but there exists none satisfying X+1 conditions. If there exists a sequence of zeroes and ones satisfying all the given conditions, then number X should be the number of all the questions asked.

Example 1

Input

```
10
5
1 2 even
3 4 odd
5 6 even
1 6 even
7 10 odd
```

Output

```
3
```

Example 2

Input

```
10
5
1 2 even
1 4 even
2 4 odd
1 10 even
3 10 even
```

Output

```
5
```