CS102A Fall 2020 Assignment 2

Problem 1. Hard Math Problem[Easy]

Description

Ahsoka must pass a very hard trial to become a Padawan (学徒). Now she gets stuck in a very hard math problem: compute the integral of a polynomial. However she has never learnt to evaluate an integral, but she knows that she can get an approximation of an integral by calculating a finite sum. Now she seeks your help.

The integral is written as the form $\int_a^b f(x)\mathrm{d}x$. $f(x)$ is a polynomial and you can compute the approximation by evaluating the value $(f(a)+f(a+\Delta x)+f(a+2\Delta x)+\cdots+f(b-\Delta x))\times\Delta x$, where $\Delta x = 0.0001$.

Input

The program receives input from `System.in`.

The input consists of 3 lines.

The first line has 2 integers a, b indicating the lower and upper bounds.

The second line has only 1 non-negative number n, indicating the degree of the polynomial (which means the polynomial is $c_1x^n + c_2x^{n-1} + \cdots + c_nx + c_{n+1}$).

The third line consists of $n+1$ **floating point numbers**, indicating the coefficients of the polynomial.

It is guaranteed that the $-10 \le a < b \le 10, 1 \le n \le 3$ and all the coefficients satisfy $-100 \le c \le 100$ with at most 3 digits.

Output

Only one number, the approxiation of the integral **using the given method**. You should **round** the answer **to the nearest tenth.**

Sample input

```
-1 10
2
1.01 -0.45 3.6
```

Sample output

```
354.3
```

Hint

Use `double` for more precise calculation.

$\int_{-1}^{10}(1.01x^2-0.45x+3.6)\mathrm{d}x\approx 354.3$

Try `System.out.printf("%.1f\n", ans);` for auto-rounding.

Problem 2. Shattered [Hard]

Description

Ahsoka is on the run. Order 66 has been executed, and now the clone troopers are hunting down Ahsoka at all costs. Ahsoka must leave the Jedi cruiser as soon as possible. The map of the Jedi cruiser is given by an $n \times m$ matrix A, and A_{ij} (the element at the i-th row, j-th column of A) can be represented by one of the following notations:

1. `W` - Wall
2. `C` - Corridor

Initially Ahsoka lies in the coordinate $(1, 1)$ of the matrix, and she can travel through the corridor to get to the exit located at the point (n, m).

Please help her escape.

Input

The first line consists of 2 positive integer n, m .

For the following n lines, each line contains m marks (`W` or `C`).

It is guaranteed that $1 \leq n, m \leq 50$ and points $(1, 1), (n, m)$ are always character `C`. It is also guaranteed that there is either **no way out** or **only one** path. There may exist some branch roads which lead to dead ends, but the branches have **maximum length of 1** and there are **no other branches** on a branch.

Output

The first line contains one word "Yes" or “No” (without quotes) indicates whether Ahsoka can successfully escape or not.

If the answer is “Yes” (without quotes), then print the coordinate of each step Ahsoka has to take in order. One coordinate per line.

Sample Input 1

```
3 3
CWW
CWW
CWC
```

Sample Output 1

```
No
```

Sample Input 2

```
3 4
CCCW
WWCW
WCCC
```

Sample Output 2

```
Yes
(1,1)
(1,2)
(1,3)
(2,3)
(3,3)
(3,4)
```

Hint

Try using `charAt()` to fetch character(s) from a `String`.

Problem 3. Harmonic Strings [Medium]

Description

A noob guitarist is trying hard to study Java. One day he comes up with a strange idea: “Since the character sequence is called `String` in Java, can it produce harmonic sounds like a guitar do?”

He wants to verify his thoughts as soon as possible, but he still has tons of bugs to fix. Fortunately, you are so kind to help him to do the calculation.

For a string s, let l be the length of the string, then the frequency produced by the string is:

$$\text{freq}(s) = \left(s_0 + s_1 l + s_2 l^2 + \cdots + s_{l-1} l^{l-1}\right) \bmod 998244353$$

Where $\bmod$ represents the modulus operation, and $s_0, \ldots, s_{l-1}$ represent the ASCII code for the characters in s respectively.

You are given n strings and you need to calculate the frequency for each string. To simplify the problem, you only need to return the **ratio** of the largest and smallest one in **rational number form** $\frac{p}{q}$, where p and q are **relatively prime (互素)**, i.e. $\gcd(p, q) = 1$.

Input

The program receives input from `System.in`.

The first line contains an integer n, where $1 < n \leq 1000$.

For the following n lines, each line contain a string s with length not exceeding 1000.

Output

Two integers p and q (which are described above), separated by a space.

Sample Input

```
2
Hello world!
d
```

Sample Output

```
7151573 1
```

Explanation

The frequencies of the two strings are 715157300 and 100 respectively, and $\frac{715157300}{100} = \frac{7151573}{1}$.

Watch out for integer overflowing!

Problem 4. Silly Calculator [Medium]

Description

Macromogic is not getting along well with his roommate recently. He decides to hack into his roommate's *CASIO fx-991CN* calculator to mess him up in the exam. The project may be huge, so he hires you to do the simple steps.

You are given a sequence containing digits and arithmetic operators (`+`, `-`, `*`, `/`). The only thing you should do is to calculate the result **from left to right, regardless of the operator precedence**. It is guaranteed that the numbers in the sequence are in the range of `int` type, and **you do not need to care if there is an overflow**. The divisor will never be 0, and if the division result is not an integer, just take the **integral part** (i.e. the maximum integer not exceeding the number). For example:

$$\begin{gathered} 1 \div 2 = 0 \\ 3 \div 2 = 1 \\ -1 \div 2 = -1 \\ -3 \div 2 = -2 \end{gathered}$$

If you can finish his task successfully, you will not get paid or receive other rewards, but you can get full marks in this assignment.

Input

The program receives input from `System.in`.

The input has only one line, containing a string as described above. It is guaranteed that the string starts with a digit.

Output

One integer, representing the calculation result.

Sample Input

```
1+2*6/5-9
```

Sample Output

```
-6
```

Explanation

$$\begin{gathered} 1 + 2 = 3 \\ 3 \times 6 = 18 \\ 18 \div 5 = 3 \\ 3 - 9 = -6 \end{gathered}$$

Problem 5. Jimmy’s Lottery [Medium]

Description

Jimmy is a boy who is interested in lottery recently. He wants to define a new kind of lottery by himself. The new kind of lottery is a set of number which is similar as the original lottery, but Jimmy has changed the rule. For all sets of number in every turn, there is a standard set which will be used to calculate the other sets' scores. For each digit in a set, if it is the same as the digit in the standard set at the same position, one gets 2 points for this digit. If the absolute difference between two number is less than 3, one gets 1 point. Otherwise one gets no points. For example, if the standard set is $(1, 2, 3)$ and your set is $(0, 9, 2)$, then your score will be $1 + 0 + 1 = 2$ points.

To make it more fun, Jimmy also set a lucky number. The lucky number can only be calculated by all digits in set. The formula for the lucky number is $\sum_{r=1}^{n} a_r \bmod q$ (for given "lucky divisor" q, and a_r represents each digit in the set). The lucky number is also counted for final score, which has the same rule as other digits (mentioned above). For example, the standard set is $(1, 2, 3)$ and your set is $(0, 9, 2)$ with given $q = 3$, then the standard lucky number is $(1 + 2 + 3) \bmod 3 = 0$, and your set's lucky number is $(0 + 9 + 2) \bmod 3 = 2$. Their difference is 2, so for lucky number you will get 1 point according to the rule. The final score for your set will be $1 + 0 + 1 + 1 = 3$.

You're a well-known programmer, so Jimmy wants you to write a program for him to calculate each set's score quickly.

Input

The program receives input from `System.in` .

The input contains several lines. The first line is the number of test cases n, the length for set m and the lucky divisor q. The next line shows the standard set. The following n lines will show all test cases, one in a line.

It's guaranteed that $1 \leq n \leq 10000$, $1 \leq m \leq 1000$, $1 \leq q \leq 10$. The range for numbers in set is $[1, 1000]$. All numbers are integers.

Output

Output the score for each set. One score in a line.

Sample Input 1

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

Sample Output 1

```
10
4
5
```

Sample Input 2

```
1 1 6
5
7
```

Sample Output 2

```
1
```

Problem 6. Easy Multiplication Plus [Bonus]

Matrix multiplication, easy peasy. Right?

How about try something more challenging? Given matrices $M_1, M_2, \ldots, M_N$, can you calculate $\left(\prod_{i=1}^{N} M_i\right)^k$?

If you do not know how to multiply matrices, please refer to your Linear Algebra textbook (or the description in *Easy Multiplication*).

Input

The program receives input from `System.in`.

The first line contains two integer N, k, denoting the number of matrices and the exponent index. Then the following lines describes the n matrices $M_1, M_2, \ldots, M_N$ in order.

For the i-th matrix M_i, the first line contains two positive integers m_i, n_i denoting that the matrix M_i is $m_i \times n_i$. Then for the next m_i lines, each line contains n_i integers, which represent the entries of the matrix.

It is guaranteed that $n_i = m_{i+1}$ for all $1 \leq i < N$ and $n_N = m_1$.

Note that the input data may be huge, please:

1. Use the provided `Reader` class for input, otherwise your program will be **very slow**!

```java
import java.io.*;
import java.util.StringTokenizer;

class Main {
  public static void main(String[] args) {
    Reader reader = new Reader(System.in);
    // Write your code below.
    // You may use `reader.nextInt()' to read integers.

  }

  // You do not necessarily understand the code below currently.
  private static class Reader {
    BufferedReader in;
    StringTokenizer tokenizer;

    public Reader(InputStream inputStream) {
      in = new BufferedReader(new InputStreamReader(inputStream));
    }

    private String next() {
      while (tokenizer == null || !tokenizer.hasMoreTokens()) {
        try {
          tokenizer = new StringTokenizer(in.readLine());
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
      return tokenizer.nextToken();
    }

    public int nextInt() {
      return Integer.parseInt(next());
    }
  }
}
```

2. Calculate the products modulo 1000000007, i.e.:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \bmod 1000000007$$

3. Use “Quick power” algorithm to calculate the matrix power:

```
The following is the pseudocode for `Quick Power' (Assume we are to calculate a^b mod m):
ans <= 1
for all binary digits d of b (from right to left):
    if d is 1 then ans <= ans * a mod m
    a = a * a mod m
return ans
```

Output

An $m_1 \times m_1$ square matrix, representing the total product. Entries in the same row are separated by a space.

Sample Input

```
2 2
2 3
1 0 1
2 1 -2
3 2
1 2
2 1
0 -1
```

Sample Output

```
5 8
32 53
```

Explanation

$$\left(\begin{pmatrix} 1 & 0 & 1 \\ 2 & 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 2 & 1 \\ 0 & -1 \end{pmatrix}\right)^2 = \begin{pmatrix} 1 & 1 \\ 4 & 7 \end{pmatrix}^2 = \begin{pmatrix} 5 & 8 \\ 32 & 53 \end{pmatrix}$$