HARBOUR.SPACE UNIVERSITY

Educational Codeforces Round 40 (Rated for Div. 2)

A. Diagonal Walking

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Mikhail walks on a 2D plane. He can go either up or right. You are given a sequence of Mikhail's moves. He thinks that this sequence is too long and he wants to make it as short as possible.

In the given sequence moving up is described by character `U` and moving right is described by character `R`. Mikhail can replace any pair of consecutive moves `RU` or `UR` with a diagonal move (described as character `D`). After that, he can go on and do some other replacements, until there is no pair of consecutive moves `RU` or `UR` left.

Your problem is to print the minimum possible length of the sequence of moves after the replacements.

Input

The first line of the input contains one integer n ($1 \le n \le 100$) — the length of the sequence. The second line contains the sequence consisting of n characters `U` and `R`.

Output

Print the minimum possible length of the sequence of moves after all replacements are done.

Examples

input	Copy
5 RUURU	

output	Copy
3	

input	Copy
17 UUURRRRUUURURUUU	

output	Copy
13	

Note

In the first test the shortened sequence of moves may be `DUD` (its length is 3).

In the second test the shortened sequence of moves can be `UUDRRRDUDDUUU` (its length is 13).

B. String Typing

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a string s consisting of n lowercase Latin letters. You have to type this string using your keyboard.

Initially, you have an empty string. Until you type the whole string, you may perform the following operation:

- add a character to the end of the string.

Besides, **at most once** you may perform one additional operation: copy the string and append it to itself.

For example, if you have to type string `abcabca`, you can type it in 7 operations if you type all the characters one by one. However, you can type it in 5 operations if you type the string `abc` first and then copy it and type the last character.

If you have to type string `aaaaaaaaa`, the best option is to type 4 characters one by one, then copy the string, and then type the remaining character.

Print the minimum number of operations you need to type the given string.

Input

The first line of the input containing only one integer number n ($1 \le n \le 100$) — the length of the string you have to type. The second line containing the string s consisting of n lowercase Latin letters.

Output

Print one integer number — the minimum number of operations you need to type the given string.

Examples

input Copy

```
7
abcabca
```

output Copy

```
5
```

input Copy

```
8
abcdefgh
```

output Copy

```
8
```

Note

The first test described in the problem statement.

In the second test you can only type all the characters one by one.

C. Matrix Walk

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a matrix A of size $x \times y$ filled with integers. For every $i \in [1..x], j \in [1..y] A_{i,j} = y(i - 1) + j$. Obviously, every integer from $[1..xy]$ occurs exactly once in this matrix.

You have traversed some path in this matrix. Your path can be described as a sequence of visited cells a_1, a_2, ..., a_n denoting that you started in the cell containing the number a_1, then moved to the cell with the number a_2, and so on.

From the cell located in i-th line and j-th column (we denote this cell as (i, j)) you can move into one of the following cells:

1. $(i + 1, j)$ — only if $i < x$;
2. $(i, j + 1)$ — only if $j < y$;
3. $(i - 1, j)$ — only if $i > 1$;
4. $(i, j - 1)$ — only if $j > 1$.

Notice that making a move requires you to go to an adjacent cell. It is not allowed to stay in the same cell. You don't know x and y exactly, but you have to find any possible values for these numbers such that you could start in the cell containing the integer a_1, then move to the cell containing a_2 (in one step), then move to the cell containing a_3 (also in one step) and so on. Can you choose x and y so that they don't contradict with your sequence of moves?

Input

The first line contains one integer number n ($1 \le n \le 200000$) — the number of cells you visited on your path (if some cell is visited twice, then it's listed twice).

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^9$) — the integers in the cells on your path.

Output

If all possible values of x and y such that $1 \le x, y \le 10^9$ contradict with the information about your path, print `NO`.

Otherwise, print `YES` in the first line, and in the second line print the values x and y such that your path was possible with such number of lines and columns in the matrix. Remember that they must be positive integers not exceeding 10^9.

Examples

input Copy

```
8
1 2 3 6 9 8 5 2
```

output Copy

```
YES
3 3
```

input Copy

```
6
1 2 1 2 5 3
```

output Copy

```
NO
```

input Copy

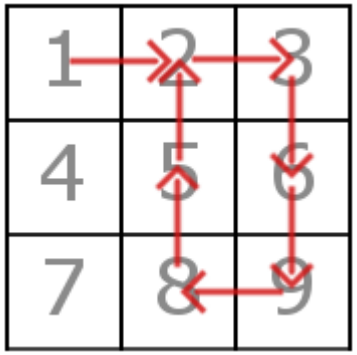
```
2
1 10
```

output Copy

```
YES
4 9
```

Note

The matrix and the path on it in the first test looks like this:



Also there exist multiple correct answers for both the first and the third examples.

D. Fight Against Traffic

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Little town Nsk consists of n junctions connected by m bidirectional roads. Each road connects two distinct junctions and no two roads connect the same pair of junctions. It is possible to get from any junction to any other junction by these roads. The distance between two junctions is equal to the minimum possible number of roads on a path between them.

In order to improve the transportation system, the city council asks mayor to build one new road. The problem is that the mayor has just bought a wonderful new car and he really enjoys a ride from his home, located near junction s to work located near junction t. Thus, he wants to build a new road in such a way that the distance between these two junctions won't decrease.

You are assigned a task to compute the number of pairs of junctions that are not connected by the road, such that if the new road between these two junctions is built the distance between s and t won't decrease.

Input

The firt line of the input contains integers n, m, s and t ($2 \le n \le 1000$, $1 \le m \le 1000$, $1 \le s, t \le n$, $s \ne t$) — the number of junctions and the number of roads in Nsk, as well as the indices of junctions where mayors home and work are located respectively. The i-th of the following m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), meaning that this road connects junctions u_i and v_i directly. It is guaranteed that there is a path between any two junctions and no two roads connect the same pair of junctions.

Output

Print one integer — the number of pairs of junctions not connected by a direct road, such that building a road between these two junctions won't decrease the distance between junctions s and t.

Examples

input Copy

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

output Copy

```
0
```

input Copy

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

output Copy

```
5
```

input Copy

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

output Copy

```
3
```

E. Water Taps

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Consider a system of n water taps all pouring water into the same container. The i-th water tap can be set to deliver any amount of water from 0 to a_i ml per second (this amount may be a real number). The water delivered by i-th tap has temperature t_i.

If for every $i \in [1, n]$ you set i-th tap to deliver exactly x_i ml of water per second, then the resulting temperature of water will be $\frac{\sum_{i=1}^{n} x_i t_i}{\sum_{i=1}^{n} x_i}$ (if $\sum_{i=1}^{n} x_i = 0$, then to avoid division by zero we state that the resulting water temperature is 0).

You have to set all the water taps in such a way that the resulting temperature is exactly T. What is the maximum amount of water you may get per second if its temperature has to be T?

Input

The first line contains two integers n and T ($1 \le n \le 200000$, $1 \le T \le 10^6$) — the number of water taps and the desired temperature of water, respectively.

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^6$) where a_i is the maximum amount of water i-th tap can deliver per second.

The third line contains n integers t_1, t_2, ..., t_n ($1 \le t_i \le 10^6$) — the temperature of water each tap delivers.

Output

Print the maximum possible amount of water with temperature exactly T you can get per second (if it is impossible to obtain water with such temperature, then the answer is considered to be 0).

Your answer is considered correct if its absolute or relative error doesn't exceed 10^{-6}.

Examples

input	Copy
2 100 3 10 50 150	

output Copy

```
6.000000000000000
```

input Copy

```
3 9
5 5 30
6 6 10
```

output Copy

```
40.000000000000000
```

input Copy

```
2 12
1 3
10 15
```

output Copy

```
1.666666666666667
```

F. Runner's Problem

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are running through a rectangular field. This field can be represented as a matrix with 3 rows and m columns. (i, j) denotes a cell belonging to i-th row and j-th column.

You start in $(2, 1)$ and have to end your path in $(2, m)$. From the cell (i, j) you may advance to:

- $(i - 1, j + 1)$ — only if $i > 1$,
- $(i, j + 1)$, or
- $(i + 1, j + 1)$ — only if $i < 3$.

However, there are n obstacles blocking your path. k-th obstacle is denoted by three integers a_k, l_k and r_k, and it forbids entering any cell (a_k, j) such that $l_k \le j \le r_k$.

You have to calculate the number of different paths from $(2, 1)$ to $(2, m)$, and print it modulo $10^9 + 7$.

Input

The first line contains two integers n and m ($1 \le n \le 10^4$, $3 \le m \le 10^{18}$) — the number of obstacles and the number of columns in the matrix, respectively.

Then n lines follow, each containing three integers a_k, l_k and r_k ($1 \le a_k \le 3$, $2 \le l_k \le r_k \le m$ - 1) denoting an obstacle blocking every cell (a_k, j) such that $l_k \le j \le r_k$. Some cells may be blocked by multiple obstacles.

Output

Print the number of different paths from $(2, 1)$ to $(2, m)$, taken modulo $10^9 + 7$. If it is impossible to get from $(2, 1)$ to $(2, m)$, then the number of paths is 0.

Example

input Copy

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

output Copy

2

G. Castle Defense

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Today you are going to lead a group of elven archers to defend the castle that is attacked by an army of angry orcs. Three sides of the castle are protected by impassable mountains and the remaining side is occupied by a long wall that is split into n sections. At this moment there are exactly a_i archers located at the i-th section of this wall. You know that archer who stands at section i can shoot orcs that attack section located at distance not exceeding r, that is all such sections j that $|i - j| \le r$. In particular, $r = 0$ means that archers are only capable of shooting at orcs who attack section i.

Denote as defense level of section i the total number of archers who can shoot at the orcs attacking this section. Reliability of the defense plan is the minimum value of defense level of individual wall section.

There is a little time left till the attack so you can't redistribute archers that are already located at the wall. However, there is a reserve of k archers that you can distribute among wall sections in arbitrary way. You would like to achieve maximum possible reliability of the defence plan.

Input

The first line of the input contains three integers n, r and k ($1 \le n \le 500\,000$, $0 \le r \le n$, $0 \le k \le 10^{18}$) — the number of sections of the wall, the maximum distance to other section archers can still shoot and the number of archers yet to be distributed along the wall. The second line contains n integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 10^9$) — the current number of archers at each section.

Output

Print one integer — the maximum possible value of defense plan reliability, i.e. the maximum possible value of minimum defense level if we distribute k additional archers optimally.

Examples

input Copy

```
5 0 6
5 4 3 4 9
```

output Copy

```
5
```

input Copy

```
4 2 0
1 2 3 4
```

output Copy

```
6
```

input Copy

```
5 1 1
2 1 2 1 2
```

output Copy

```
3
```

H. Path Counting

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a rooted tree. Let's denote $d(x)$ as depth of node x: depth of the root is 1, depth of any other node x is $d(y) + 1$, where y is a parent of x.

The tree has the following property: every node x with $d(x) = i$ has exactly a_i children. Maximum possible depth of a node is n, and $a_n = 0$.

We define f_k as the number of unordered pairs of vertices in the tree such that the number of edges on the simple path between them is equal to k.

Calculate f_k modulo $10^9 + 7$ for every $1 \le k \le 2n$ - 2.

Input

The first line of input contains an integer n $(2 \le n \le 5\,000)$ — the maximum depth of a node.

The second line of input contains n - 1 integers $a_1, a_2, \ldots, a_{n-1}$ $(2 \le a_i \le 10^9)$, where a_i is the number of children of every node x such that $d(x) = i$. Since $a_n = 0$, it is not given in the input.

Output

Print $2n$ - 2 numbers. The k-th of these numbers must be equal to f_k modulo $10^9 + 7$.

Examples

input Copy

```
4
2 2 2
```

output Copy

```
14 19 20 20 16 16
```

input Copy

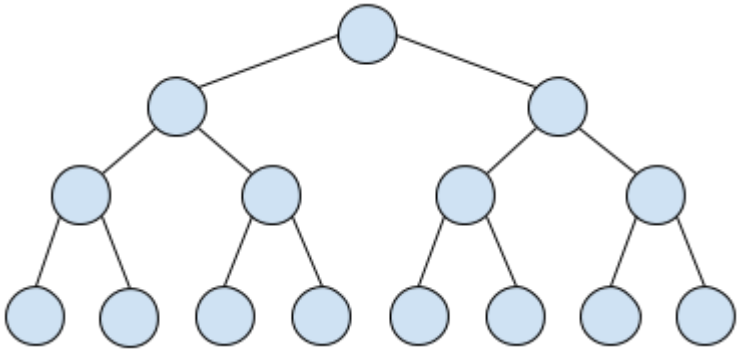
```
3
2 3
```

output Copy

```
8 13 6 9
```

Note

This the tree from the first sample:



I. Yet Another String Matching Problem

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Suppose you have two strings s and t, and their length is equal. You may perform the following operation any number of times: choose two different characters c_1 and c_2, and replace every occurence of c_1 in both strings with c_2. Let's denote the *distance* between strings s and t as the minimum number of operations required to make these strings equal. For example, if s is `abcd` and t is `ddcb`, the *distance* between them is 2 — we may replace every occurence of `a` with `b`, so s becomes `bbcd`, and then we may replace every occurence of `b` with `d`, so both strings become `ddcd`.

You are given two strings S and T. For every substring of S consisting of $|T|$ characters you have to determine the *distance* between this substring and T.

Input

The first line contains the string S, and the second — the string T ($1 \le |T| \le |S| \le 125000$). Both strings consist of lowercase Latin letters from `a` to `f`.

Output

Print $|S| - |T| + 1$ integers. The i-th of these integers must be equal to the *distance* between the substring of S beginning at i-th index with length $|T|$ and the string T.

Example

input	Copy
abcdefa ddcb	

output	Copy
2 3 3 3	