Day 3. Dynamic programming

1A. Number of Grasshopper Paths

1 second, 256 megabytes

There is a grass field consisting of n grass cells. Initially, a grasshopper is staying at the cell 1. It can jump to the right by 1, 2, 3, ..., k (i.e. it can jump to any cell to the right that is not further than k from the current cell).

Your task is to calculate the number of different paths from the cell 1 to the cell n.

Since the answer can be very large, print it modulo $10^9 + 7$ (1000000007).

Input
The only line of the input contains two integers n and k ($1 \le k < n \le 1000$) — the number of cells in the grass field and the maximum jump length, respectively.

Output
Print one integer — the number of different paths from the cell 1 to the cell n.

Since the answer can be very large, print it modulo $10^9 + 7$ (1000000007).

input
5 2
output
5

input
1000 215
output
801061711

1B. Path in the Table

2 seconds, 256 megabytes

There is a table of size $n \times m$. There is $a_{i,j}$ coins in the cell (i, j) of this table. You start in the cell $(1, 1)$ and want to reach the cell (n, m). You can only go right (from the cell (i, j) to the cell $(i, j + 1)$) or down (from the cell (i, j) to the cell $(i + 1, j)$). Note that you **cannot** move right from the column m or go down from the row n (i.e. you cannot move outside the table).

Your task is to calculate the **maximum** number of coins you can get among all possible paths from $(1, 1)$ to (n, m).

Input
The first line of the input contains two integers n and m ($1 \le n, m \le 500$) — the number of rows and the number of columns, respectively.

The next n lines contain m integers each. The j-th number in the i-th line is $a_{i,j}$ ($-10^6 \le a_{i,j} \le 10^6$) — the number of coins in the cell (i, j).

Output
Print one integer — the **maximum** number of coins you can get among all possible paths from $(1, 1)$ to (n, m).

input
3 3 2 4 9 1 3 2 11 3 1
output
18

input
4 4 8 5 -3 -10 3 -9 -8 -10 2 -1 -3 -8 -3 1 9 3
output
25

1C. One-Dimensioanl Sea Battle

1 second, 512 megabytes

You are given a one-dimensional Sea Battle field of length n (i.e. it consists of n cells in a row).

Your task is to calculate the number of different ways to place **any** amount of single-deck and double-deck battleships (i.e. ships consisting of one cell and ships consisting of two consecutive cells).

By the rules of Sea Battle, there should be **at least one empty cell** between any pair of battleships.

Since the answer can be very large, print it modulo $10^9 + 7$ (1000000007).

Input
The only line of the input contains one integer n ($1 \le n \le 10^5$) — the length of the field.

Output
Print one integer — the number of ways to place **any** amount of single-deck and double-deck battleships.

Since the answer can be very large, print it modulo $10^9 + 7$ (1000000007).

input
1
output
2

input
3
output
7

input
20

output
223317

input
51423
output
204721646

1D. Platforms

1 second, 256 megabytes

You are playing a computer game. There are n platforms in a row. The height of the i-th platform is h_i.

You are standing on the first platform. Your target is to reach the n-th platform.

You can make one of the two following actions:

- Jump from the current platform i to the platform $i+1$. This action takes $|h_i - h_{i+1}|$ energy.
- Jump from the current platform i to the platform $i+2$. This action takes $3|h_i - h_{i+2}|$ energy. Of course, you **can not** make that action from the platform $n-1$ (otherwise your character will fall off the field and die).

Your task is to calculate the minimum amount of energy required to reach the platform n from the platform 1.

Input
The first line of the input contains one integer n ($1 \le n \le 10^5$) — the number of platforms.

The second line of the input contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 10^9$), where h_i is the height of the i-th platform.

Output
Print one integer — the minimum amount of energy required to reach the platform n from the platform 1.

input
3 1 5 10
output
9

input
3 1 5 2
output
3

input
4 1 10 5 4
output
13

1E. Path in the Table - 3

1 second, 256 megabytes

There is a table with n rows and m columns.

From any cell (i, j) you can move either to cell $(i, j+1)$ or to cell $(i+1, j)$ but you can't move outside the table.

The problem is that some cells in the table are blocked, i.e. it isn't possible to move to this cell from any other cell.

You want to find the number of paths from cell $(1, 1)$ to cell (n, m). Of course, each path can't contain blocked cells.

As the described number can be rather large, print the remainder after dividing it by 1000000007 ($10^9 + 7$).

Input
The first line of the input contains two integers n and m ($1 \le n, m \le 500$) — the number of rows and the number of columns, respectively.

The next n lines contain m characters each. The j-th character in the i-th line is `'#'` if the cell is blocked and `'.'` otherwise.

It's guaranteed that cells $(1, 1)$ and (n, m) aren't blocked.

Output
Print a single integer — the answer to the problem.

input
3 3 .#. ..# ...
output
2

input
5 5 ....
output
70

2A. Removing brackets

2 seconds, 256 megabytes

Given a string made up of round, square and curly brackets. Find a subsequence of maximal length that forms the correct bracket sequence.

Input
The sole line of the input contains a string of round, square and curly brackets. The length of the string does not exceed 100 characters.

Output
The sole line of the output should contain the maximum length of a string that is a correct bracket sequence and can be obtained from the original string by deleting some characters.

input
([)]
output
2

input
{([(]{)})]
output
6

2B. Independent set of maximum weight

1 second, 256 megabytes

Given a tree of n vertices. The vertex i has weight w_i. Independent set is a set of vertices in which no two vertices are connected by an edge. For a given tree find an independent set with the maximum total weight.

Input
The first line contains the integer n, the number of tree nodes ($1 \le n \le 10^5$). The second line contains n integers w_i ($1 \le w_i \le 1000$). The next $n - 1$ lines describe the edges of the tree. Each line contains two numbers, the indices of the vertices that the edge connects. It is guaranteed that the described graph is a tree.

Output
Print one number, the maximum total weight of the vertices of the independent set.

input
6 2 3 1 1 3 1 1 2 1 6 4 2 6 5 6 3
output
7

2C. Probabilistic Candies

2 seconds, 1024 megabytes

There are n dishes in a row, on top of the i-th of them are a_i candies ($1 \le a_i \le 3$).

You perform the following move until all candies are eaten: roll a die with n faces, the i-th face contains a number i. Let the outcome is the face with the number i. Then, if there are some candies left on the i-th dish, eat one of them, otherwise do nothing.

Your task is to calculate the expected number of rolls required to eat all the candies.

Input
The first line of the input contains one integer n ($1 \le n \le 300$) — the number of dishes.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 3$), where a_i is the number of candies on the i-th dish.

Output
Print one real number — the expected number of rolls required to eat all the candies. Your answer will be considered correct if its absolute error is at most 10^{-9}.

input
3 1 1 1
output
5.5000000000

input
1 3
output
3.0000000000

input
2 1 2
output
4.5000000000

input
10 1 3 2 3 3 2 3 2 1 3
output
54.4806445749

Consider the first example. The expected number of rolls before the first candy is eaten is 1. After that, the expected number of rolls before the second candy is eaten is 1.5. After that, the expected number of operations before the third candy is eaten is 3. Thus, the expected total number of operations is $1 + 1.5 + 3 = 5.5$.

2D. Potions Combining

1 second, 256 megabytes

You are an ancient alchemist and currently experimenting with potions combining. More precisely, there are n potions staying in a row. The toxicity of the i-th potion is a_i.

In one move, you can combine two adjacent potions i and $i + 1$. The toxicity of this move is $a_i + a_j$. After that move, potions i and $i + 1$ are removed and replaced with on potion with the toxicity $a_i + a_j$.

Your task is to calculate the minimum possible total toxicity you can reach to combine all n potions into one.

Input
The first line of the input contains one integer n ($2 \le n \le 400$) — the number of potions.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the toxicity of the i-th potion.

Output
Print one integer — the minimum possible total toxicity you can reach to combine all n potions into one.

input
4 10 20 30 40
output
190

input
5 10 10 10 10 10

output

```
120
```

input

```
3
1000000000 1000000000 1000000000
```

output

```
5000000000
```

input

```
6
7 6 8 6 1 1
```

output

```
68
```

Consider the first example. The following sequence of combines is the optimal:

1. $(10, 20, 30, 40) \to (30, 30, 40)$;
2. $(30, 30, 40) \to (60, 40)$;
3. $(60, 40) \to (100)$.

Consider the second example. The following sequence of combines is the optimal:

1. $(10, 10, 10, 10, 10) \to (20, 10, 10, 10)$;
2. $(20, 10, 10, 10) \to (20, 20, 10)$;
3. $(20, 20, 10) \to (20, 30)$;
4. $(20, 30) \to (50)$.

2E. Tower Building

1 second, 256 megabytes

You are given n blocks. The i-th block has its weight w_i, solidness s_i and value v_i.

Your task is to choose some blocks and build the tower from them, **maximizing** its cost. In other words, you choose some subset of blocks, rearrange them somehow and stack one on another. For each block, the sum of weights (w_i) of the blocks above it should be not greater than its solidness s_i.

Input

The first line of the input contains one integer n ($1 \le n \le 1000$) — the number of blocks.

The next n lines describe blocks. The i-th of them contains three integers w_i, s_i and v_i ($1 \le w_i, s_i \le 10^4; 1 \le v_i \le 10^9$) — the weight, the solidness and the cost of the i-th block, respectively.

Output

Print one integer — the maximum cost of the tower you can reach satisfying the conditions from the problem statement.

input

```
3
2 2 20
2 1 30
3 1 40
```

output

```
50
```

input

```
4
1 2 10
3 1 10
2 4 10
1 6 10
```

output

```
40
```

input

```
5
1 10000 1000000000
1 10000 1000000000
1 10000 1000000000
1 10000 1000000000
1 10000 1000000000
```

output

```
5000000000
```

input

```
8
9 5 7
6 2 7
5 7 3
7 8 8
1 9 6
3 3 3
4 1 7
4 5 5
```

output

```
22
```

2F. Restricted Permutations

1 second, 256 megabytes

You are given an integer n and a string s consisting of $n - 1$ characters '<' and '>'.

Your task is to calculate the number of permutations of length n that satisfy the given string s. The permutation $p_1, p_2, \ldots, p_n$ satisfies the string s if for all i from 1 to $n - 1$ the following condition is satisfied: if $s_i =$ '<' then $p_i < p_{i+1}$, otherwise (if $s_i =$ '>') then $p_i > p_{i+1}$.

Recall that the permutation of length n is such an array of length n that contains each integer from 1 to n **exactly once**.

Input

The first line of the input contains one integer n ($2 \le n \le 3000$) — the length of the permutation.

The second line of the input contains the string s consisting of $n - 1$ characters '<' and '>'.

Output

Print one integer — the number of permutations of length n that satisfy the given string s. Since the answer can be very large, print it modulo $10^9 + 7$ (1000000007).

input

```
4
<><
```

output

```
5
```

input
5 <<<<
output
1

input
20 >>>><>>><><>><<>>
output
217136290

Consider the first example. Suitable permutations are:

1. $[1, 3, 2, 4]$;
2. $[1, 4, 2, 3]$;
3. $[2, 3, 1, 4]$;
4. $[2, 4, 1, 3]$;
5. $[3, 4, 1, 2]$.

3A. Runners

3 seconds, 256 megabytes

Byteland consists of $n + 1$ cities located along a straight road. Cities are numbered from 0 to n in order of increasing coordinates. The capital is city 0 and it is located at the point with the coordinate 0. City i is located at x_i. Phones have not yet been invented, therefore, in case of emergency in each city there is a special runner. If an emergency occurs, a runner in city i will prepare for a_i seconds, after that he will run towards the capital, spending b_i seconds for one kilometer. When a runner reaches another city, he can either run past it or go into the city and give the message to the local runner (in this case, the second runner also prepares, and then runs towards the capital).

For each city find the minimum time for a message from it to be delivered to the capital.

Input

The first line contains the number n ($1 \le n \le 10^6$). Each of the following n lines contains a triple of numbers x_i, a_i, b_i ($1 \le x_i \le 10^6$, $0 \le a_i \le 10^6$, $1 \le b_i \le 10^6$).

Output

Print n numbers — the minimum time for cities 1, 2, 3, ..., n.

input
4 2 5 2 3 1 4 6 0 10 10 10 1
output
9 13 43 20

Statement is not available on English language

3C. High Probability Cast

2 seconds, 256 megabytes

A mage knows n spells, the i-th of which, when casted, deals random damage (not necessarily integer), uniformly distributed from a_i to b_i. There are m monsters dwelling in the world, and to kill the j-th of them it is required to deal him at least x_j damage. Unfortunately, monsters are so fast that the mage has time to cast only a single spell while fighting each of the monsters, before being killed by that monster. Which spell should be chosen to destroy each of the monsters, so that the probability of killing that monster would be maximized?

Input

The first line contains a single integer n ($1 \le n \le 200000$) — the number of spells.

The next n lines describe spells. Each of them contains two integers a_i and b_i ($1 \le a_i \le b_i \le 10^9$) — the minimal and maximal damage the i-th spell can deal.

The next line contains a single integer m ($1 \le m \le 200000$) — the number of monsters.

The next line contains m integers x_j separated by a space ($1 \le x_j \le 10^9$) — the minimal amount of damage required to destroy the j-th monster.

Output

Output m integers separated by a space, the j-th of which should be equal to the number of spell which should be used to destroy the j-th monster. The spells are numbered from one in the order they are listed in the input. If there are many spells which give the maximal probability of destroying some monster, you can output any of these spells.

input
2 1 10 4 8 4 3 6 7 11
output
2 2 1 1

input
2 2 5 7 9 5 10 8 6 3 1
output
2 2 2 2 2

3D. Good Inflation

1 second, 256 megabytes

The competition is drawing to a close, and The Team could really use another balloon to help them come out on top...good thing they're no chumps, and thought of this in advance! Before the contest started, they instructed an accomplice to go to the local balloon shop, buy an extra balloon, and deliver it to them at the conclusion of the event. It is true that their balloons then won't match up with the scoreboard's record of which problems they've solved - but, in the end, it's the balloons that count! In fact, The Team knows that, the larger their extra balloon, the more their opponents (and the judges) will be intimidated. They're not the best ACM-ICPC team in the world for nothing!

However, this balloon shop is rather strange. The accomplice is given an empty balloon (a balloon with size 0), and told that he can tie it closed and keep it after exactly N ($1 \leq N \leq 10^6$) minutes. In the meantime, during each minute, an inflation offer is available. If offer i is taken, then, at the start of the ith minute, the balloon's size will be increased by a_i ($0 \leq a_i \leq 10^6$). However, since the balloon cannot be closed until the end, its air will leak out at a constant rate, which depends on the gas that was used - in particular, its size will immediately start to decrease at a rate of d_i ($0 \leq d_i \leq 10^6$) per minute, until either another offer is taken, or the balloon deflates completely (its size can never become negative, of course).

The Team will not be happy if they don't receive the largest balloon possible. Unfortunately, their accomplice happens to be...you. Can you figure out the maximal size that the balloon can have at the start of minute $N + 1$, before it's too late?

Input
First line: 1 integer, N

Next N lines: 2 integers, a_i and d_i, for $i = 1..N$

Output
1 integer, the maximal size which the balloon can have at the start of minute $N + 1$.

input
5 2 3 10 2 0 1 5 4 1 10
output
5

The best option is to take only the offers at minutes 2 and 3. At the start of minute 2, the balloon will be inflated to size 10, and will deflate to size 8 by the start of minute 3. At that point, the balloon's size will not change, but its deflation rate will change to 1. As a result, by the start of minute 6, its size will be 5.

Note that, if the offer at minute 1 is additionally taken, the balloon will simply be inflated to size 2 before deflating back to size 0 by the start of minute 2 - therefore, this will not change the outcome.

Also note that, if the offer at minute 4 is additionally taken, the balloon will inflate to size 12 at the start of minute 4, but its deflation rate of 4 will result in a size of 4 at the start of minute 6, which is not optimal.

3E. Quadratic segmentation - 1

1 second, 256 megabytes

You are standing on an endless straight line. It is allowed to place no more than k segments on this line. The cost of placing a segment is squarely proportional to the length of the segment. Formally, the cost to place a segment $[l, r]$ is equal to $(r - l)^2$.

There are exactly n integer points on the line that must be covered by at least one segment. What is the minimum total cost you have to spend in order to cover all those points?

A point x is said to be covered by a segment $[l, r]$ if and only if $l \leq x \leq r$. In other words, the point x should touch the segment $[l, r]$ or should be completely contained within it.

Input
First line of the input contains two integers n and k — the number of points and the maximum number of segments you can place on the line ($1 \leq n \leq 10^4$, $1 \leq k \leq min(n, 100)$).

Second line contains n space-separated integers $x_1, \ldots, x_n$ — coordinates of the points that must be covered by at least one segment ($1 \leq x_i \leq 10^6$).

Output
Print a single integer — the minimum total cost to cover all given points.