Educational Codeforces Round 59 (Rated for Div. 2)

A. Digits Sequence Dividing

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

You are given a sequence s consisting of n digits from 1 to 9.

You have to divide it into **at least two** segments (segment — is a consecutive sequence of elements) (in other words, you have to place separators between some digits of the sequence) in such a way that **each element belongs to exactly one segment** and if the resulting division will be represented as an integer numbers sequence then each next element of this sequence will be **strictly greater** than the previous one.

More formally: if the resulting division of the sequence is $t_1, t_2, \ldots, t_k$, where k is the number of element in a division, then for each i from 1 to $k-1$ the condition $t_i < t_{i+1}$ (using **numerical** comparing, it means that the integer representations of strings are compared) should be satisfied.

For example, if $s = 654$ then you can divide it into parts $[6, 54]$ and it will be suitable division. But if you will divide it into parts $[65, 4]$ then it will be bad division because $65 > 4$. If $s = 123$ then you can divide it into parts $[1, 23]$, $[1, 2, 3]$ but not into parts $[12, 3]$.

Your task is to find **any** suitable division for each of the q independent queries.

Input

The first line of the input contains one integer q ($1 \le q \le 300$) — the number of queries.

The first line of the i-th query contains one integer number n_i ($2 \le n_i \le 300$) — the number of digits in the i-th query.

The second line of the i-th query contains one string s_i of length n_i consisting only of digits from 1 to 9.

Output

If the sequence of digits in the i-th query cannot be divided into **at least two** parts in a way described in the problem statement, print the single line "NO" for this query.

Otherwise in the first line of the answer to this query print "YES", on the second line print k_i — the number of parts in your division of the i-th query sequence and in the third line print k_i strings $t_{i,1}, t_{i,2}, \ldots, t_{i,k_i}$ — your division. Parts should be printed in order of the initial string digits. It means that if you write the parts one after another without changing their order then you'll get the string s_i.

See examples for better understanding.

Example

input
4 6 654321 4 1337 2 33 4 2122
output
YES 3 6 54 321 YES 3 1 3 37 NO YES 2 21 22

B. Digital root

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

output: standard output

Today at the lesson of mathematics, Petya learns about the digital root.

The digital root of a non-negative integer is the single digit value obtained by an iterative process of summing digits, on each iteration using the result from the previous iteration to compute a digit sum. The process continues until a single-digit number is reached.

Let's denote the digital root of x as $S(x)$. Then $S(5) = 5$, $S(38) = S(3 + 8 = 11) = S(1 + 1 = 2) = 2$, $S(10) = S(1 + 0 = 1) = 1$.

As a homework Petya got n tasks of the form: find k-th positive number whose digital root is x.

Petya has already solved all the problems, but he doesn't know if it's right. Your task is to solve all n tasks from Petya's homework.

Input
The first line contains a single integer n ($1 \le n \le 10^3$) — the number of tasks in Petya's homework. The next n lines contain two integers k_i ($1 \le k_i \le 10^{12}$) and x_i ($1 \le x_i \le 9$) — i-th Petya's task in which you need to find a k_i-th positive number, the digital root of which is x_i.

Output
Output n lines, i-th line should contain a single integer — the answer to the i-th problem.

Example

input
3 1 5 5 2 3 1
output
5 38 19

C. Brutality

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

You are playing a new famous fighting game: Kortal Mombat XII. You have to perform a brutality on your opponent's character.

You are playing the game on the new generation console so your gamepad have 26 buttons. Each button has a single lowercase Latin letter from 'a' to 'z' written on it. All the letters on buttons are pairwise distinct.

You are given a sequence of hits, the i-th hit deals a_i units of damage to the opponent's character. To perform the i-th hit you have to press the button s_i on your gamepad. Hits are numbered from 1 to n.

You know that if you press some button **more than** k times **in a row** then it'll break. You cherish your gamepad and don't want to break any of its buttons.

To perform a brutality you have to land some of the hits of the given sequence. **You are allowed to skip any of them, however changing the initial order of the sequence is prohibited**. The total damage dealt is the sum of a_i over all i for the hits which weren't skipped.

Note that if you skip the hit then the counter of consecutive presses the button won't reset.

Your task is to skip some hits to deal the **maximum** possible total damage to the opponent's character and not break your gamepad buttons.

Input
The first line of the input contains two integers n and k ($1 \le k \le n \le 2 \cdot 10^5$) — the number of hits and the maximum number of times you can push the same button in a row.

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 damage of the i-th hit.

The third line of the input contains the string s consisting of exactly n lowercase Latin letters — the sequence of hits (each character is the letter on the button you need to press to perform the corresponding hit).

Output
Print one integer dmg — the **maximum** possible damage to the opponent's character you can deal without breaking your gamepad buttons.

Examples

input

7 3 1 5 16 18 7 2 10 baaaaca
output
54

input
5 5 2 4 1 3 1000 aaaaa
output
1010

input
5 4 2 4 1 3 1000 aaaaa
output
1009

input
8 1 10 15 2 1 4 8 15 16 qqwweerr
output
41

input
6 3 14 18 9 19 2 15 cccccc
output
52

input
2 1 10 10 qq
output
10

Note

In the first example you can choose hits with numbers $[1, 3, 4, 5, 6, 7]$ with the total damage $1 + 16 + 18 + 7 + 2 + 10 = 54$.

In the second example you can choose all hits so the total damage is $2 + 4 + 1 + 3 + 1000 = 1010$.

In the third example you can choose all hits expect the third one so the total damage is $2 + 4 + 3 + 1000 = 1009$.

In the fourth example you can choose hits with numbers $[2, 3, 6, 8]$. Only this way you can reach the maximum total damage $15 + 2 + 8 + 16 = 41$.

In the fifth example you can choose only hits with numbers $[2, 4, 6]$ with the total damage $18 + 19 + 15 = 52$.

In the sixth example you can change either first hit or the second hit (it does not matter) with the total damage 10.

D. Compression

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a binary matrix A of size $n \times n$. Let's denote an x-compression of the given matrix as a matrix B of size $\frac{n}{x} \times \frac{n}{x}$ such that for every $i \in [1, n], j \in [1, n]$ the condition $A[i][j] = B[\lceil \frac{i}{x} \rceil][\lceil \frac{j}{x} \rceil]$ is met.

Obviously, x-compression is possible only if x divides n, but this condition is not enough. For example, the following matrix of size 2×2 does not have any 2-compression:

$$01$$
$$10$$

For the given matrix A, find maximum x such that an x-compression of this matrix is possible.

Note that the input is given in compressed form. But even though it is compressed, you'd better use fast input.

Input
The first line contains one number n ($4 \le n \le 5200$) — the number of rows and columns in the matrix A. It is guaranteed that n is divisible by 4.

Then the representation of matrix follows. Each of n next lines contains $\frac{n}{4}$ one-digit hexadecimal numbers (that is, these numbers can be represented either as digits from 0 to 9 or as uppercase Latin letters from A to F). Binary representation of each of these numbers denotes next 4 elements of the matrix in the corresponding row. For example, if the number B is given, then the corresponding elements are `1011`, and if the number is 5, then the corresponding elements are `0101`.

Elements are not separated by whitespaces.

Output
Print one number: maximum x such that an x-compression of the given matrix is possible.

Examples

input
8 E7 E7 E7 00 00 E7 E7 E7
output
1

input
4 7 F F F
output
1

Note
The first example corresponds to the matrix:

$$\begin{matrix} 11100111 \\ 11100111 \\ 11100111 \\ 00000000 \\ 00000000 \\ 11100111 \\ 11100111 \\ 11100111 \end{matrix}$$

It is easy to see that the answer on this example is 1.

E. Vasya and Binary String

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

Vasya has a string s of length n consisting only of digits `0` and `1`. Also he has an array a of length n.

Vasya performs the following operation until the string becomes empty: choose some **consecutive** substring of **equal characters**, erase it from the string and glue together the remaining parts (any of them can be empty). For example, if he erases substring `111` from string `1`**`111`**`10` he will get the string `110`. Vasya gets a_x points for erasing substring of length x.

Vasya wants to maximize his total points, so help him with this!

Input
The first line contains one integer n ($1 \le n \le 100$) — the length of string s.

The second line contains string s, consisting only of digits `0` and `1`.

The third line contains n integers $a_1, a_2, \ldots a_n$ ($1 \le a_i \le 10^9$), where a_i is the number of points for erasing the substring of length i.

Output
Print one integer — the maximum total points Vasya can get.

Examples

input
7 1101001 3 4 9 100 1 2 3
output
109

input
5 10101 3 10 15 15 15
output
23

Note
In the first example the optimal sequence of erasings is: 11**0**1001 $\rightarrow$ 111**00**1 $\rightarrow$ 111**0**1 $\rightarrow$ **1111** $\rightarrow \varnothing$.

In the second example the optimal sequence of erasings is: 10**1**01 $\rightarrow$ 1**00**1 $\rightarrow$ **11** $\rightarrow \varnothing$.

F. Vasya and Endless Credits

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya wants to buy himself a nice new car. Unfortunately, he lacks some money. Currently he has exactly 0 burles.

However, the local bank has n credit offers. Each offer can be described with three numbers a_i, b_i and k_i. Offers are numbered from 1 to n. If Vasya takes the i-th offer, then the bank gives him a_i burles at the beginning of the month and then Vasya pays bank b_i burles at the end of each month for the next k_i months (including the month he activated the offer). **Vasya can take the offers any order he wants**.

Each month Vasya can take no more than one credit offer. **Also each credit offer can not be used more than once**. Several credits can be active at the same time. It implies that Vasya pays bank the sum of b_i over all the i of active credits at the end of each month.

Vasya wants to buy a car in the middle of some month. He just takes all the money he currently has and buys the car of that exact price.

Vasya don't really care what he'll have to pay the bank back after he buys a car. He just goes out of the country on his car so that the bank can't find him anymore.

What is the maximum price that car can have?

Input
The first line contains one integer n ($1 \le n \le 500$) — the number of credit offers.

Each of the next n lines contains three integers a_i, b_i and k_i ($1 \le a_i, b_i, k_i \le 10^9$).

Output
Print one integer — the maximum price of the car.

Examples

input
4 10 9 2 20 33 1 30 115 1 5 3 2
output
32

input
3 40 1 2 1000 1100 5 300 2 1
output

1337

Note

In the first example, the following sequence of offers taken is optimal: 4 $\rightarrow$ 3.

The amount of burles Vasya has changes the following way: 5 $\rightarrow$ 32 $\rightarrow$ -86 $\rightarrow$ .... He takes the money he has in the middle of the second month (32 burles) and buys the car.

The negative amount of money means that Vasya has to pay the bank that amount of burles.

In the second example, the following sequence of offers taken is optimal: 3 $\rightarrow$ 1 $\rightarrow$ 2.

The amount of burles Vasya has changes the following way: 0 $\rightarrow$ 300 $\rightarrow$ 338 $\rightarrow$ 1337 $\rightarrow$ 236 $\rightarrow$ -866 $\rightarrow$ ....

G. Vasya and Maximum Profit

time limit per test: 3.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya got really tired of these credits (from problem F) and now wants to earn the money himself! He decided to make a contest to gain a profit.

Vasya has n problems to choose from. They are numbered from 1 to n. The difficulty of the i-th problem is d_i. Moreover, the problems are given in the increasing order by their difficulties. **The difficulties of all tasks are pairwise distinct**. In order to add the i-th problem to the contest you need to pay c_i burles to its author. For each problem in the contest Vasya gets a burles.

In order to create a contest he needs to choose a consecutive subsegment of tasks.

So the total earnings for the contest are calculated as follows:

- if Vasya takes problem i to the contest, he needs to pay c_i to its author;
- for each problem in the contest Vasya gets a burles;
- let $gap(l, r) = \max\limits_{l \le i < r} (d_{i+1} - d_i)^2$. If Vasya takes all the tasks with indices from l to r to the contest, he also needs to pay $gap(l, r)$. If $l = r$ then $gap(l, r) = 0$.

Calculate the maximum profit that Vasya can earn by taking a consecutive segment of tasks.

Input

The first line contains two integers n and a ($1 \le n \le 3 \cdot 10^5$, $1 \le a \le 10^9$) — the number of proposed tasks and the profit for a single problem, respectively.

Each of the next n lines contains two integers d_i and c_i ($1 \le d_i, c_i \le 10^9, d_i < d_{i+1}$).

Output

Print one integer — maximum amount of burles Vasya can earn.

Examples

input
5 10 1 15 5 3 6 11 7 2 11 22
output
13

input
3 5 1 8 2 19 3 11
output
0