Educational Codeforces Round 110 (Rated for Div. 2)

A. Fair Playoff

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

Four players participate in the playoff tournament. The tournament is held according to the following scheme: the first player will play with the second, and the third player with the fourth, then the winners of the pairs will play in the finals of the tournament.

It is known that in a match between two players, the one whose skill is greater will win. The skill of the i-th player is equal to s_i and all skill levels are pairwise different (i. e. there are no two identical values in the array s).

The tournament is called **fair** if the two players with the highest skills meet in the finals.

Determine whether the given tournament is **fair**.

Input
The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

A single line of test case contains four integers s_1, s_2, s_3, s_4 ($1 \le s_i \le 100$) — skill of the players. It is guaranteed that all the numbers in the array are different.

Output
For each testcase, output YES if the tournament is **fair**, or NO otherwise.

Example

input
4 3 7 9 5 4 5 6 9 5 3 8 1 6 5 3 2
output
YES NO YES NO

Note
Consider the example:

1. in the first test case, players 2 and 3 with skills 7 and 9 advance to the finals;
2. in the second test case, players 2 and 4 with skills 5 and 9 advance to the finals. The player with skill 6 does not advance, but the player with skill 5 advances to the finals, so the tournament is not fair;
3. in the third test case, players 1 and 3 with skills 5 and 8 advance to the finals;
4. in the fourth test case, players 1 and 3 with skills 6 and 3 advance to the finals. The player with skill 5 does not advance, but the player with skill 3 advances to the finals, so the tournament is not fair.

B. Array Reodering

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

You are given an array a consisting of n integers.

Let's call a pair of indices i, j **good** if $1 \le i < j \le n$ and $\gcd(a_i, 2a_j) > 1$ (where $\gcd(x, y)$ is the greatest common divisor of x and y).

Find the maximum number of **good** index pairs if you can reorder the array a in an arbitrary way.

Input
The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of the test case contains a single integer n ($2 \le n \le 2000$) — the number of elements in the array.

The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^5$).

It is guaranteed that the sum of n over all test cases does not exceed 2000.

Output
For each test case, output a single integer — the maximum number of **good** index pairs if you can reorder the array a in an arbitrary way.

Example

input
3 4 3 6 5 3 2 1 7 5 1 4 2 4 1
output
4 0 9

Note
In the first example, the array elements can be rearranged as follows: $[6, 3, 5, 3]$.

In the third example, the array elements can be rearranged as follows: $[4, 4, 2, 1, 1]$.

C. Unstable String

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

You are given a string s consisting of the characters 0, 1, and ?.

Let's call a string **unstable** if it consists of the characters `0` and `1` and any two adjacent characters are different (i. e. it has the form `010101...` or `101010...`).

Let's call a string **beautiful** if it consists of the characters `0`, `1`, and `?`, and you can replace the characters `?` to `0` or `1` (for each character, the choice is independent), so that the string becomes **unstable**.

For example, the strings `0??10`, `0`, and `???` are beautiful, and the strings `00` and `?1??1` are not.

Calculate the number of beautiful contiguous substrings of the string s.

Input
The first line contains a single integer t ($1 \le t \le 10^4$) — number of test cases.

The first and only line of each test case contains the string s ($1 \le |s| \le 2 \cdot 10^5$) consisting of characters `0`, `1`, and `?`.

It is guaranteed that the sum of the string lengths over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case, output a single integer — the number of **beautiful** substrings of the string s.

Example

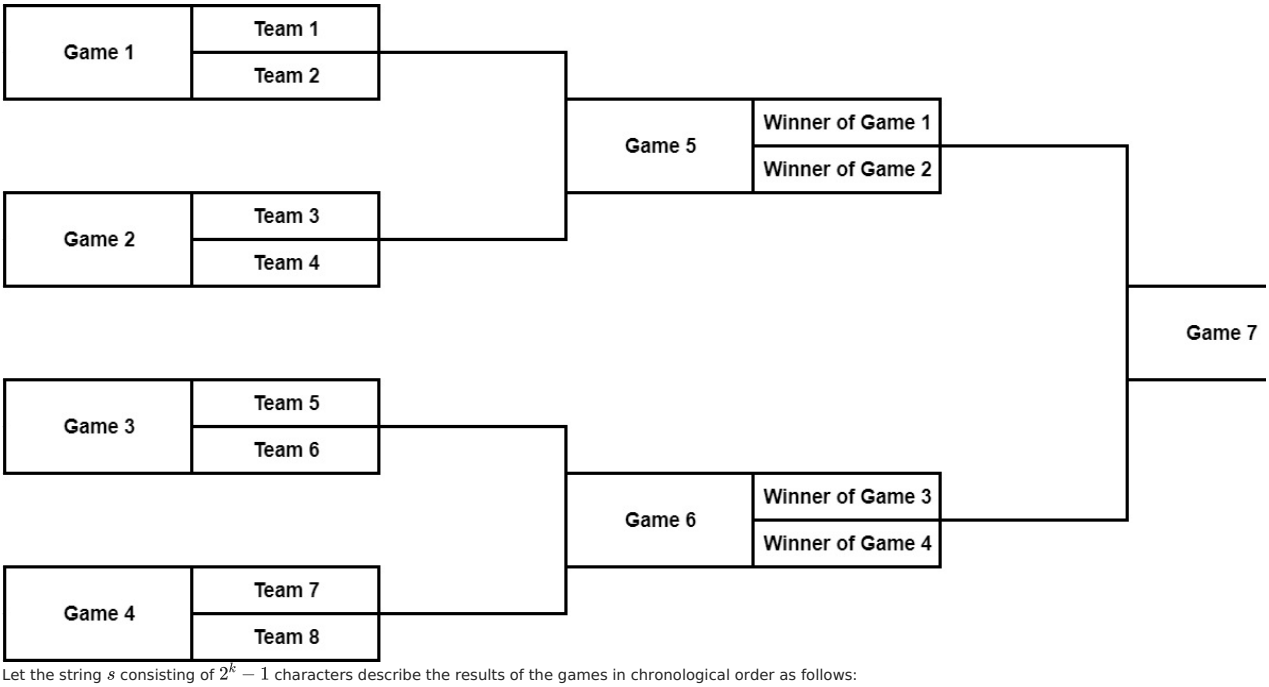
input
3 0?10 ??? ?10??1100
output
8 6 25

D. Playoff Tournament

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

2^k teams participate in a playoff tournament. The tournament consists of $2^k - 1$ games. They are held as follows: first of all, the teams are split into pairs: team 1 plays against team 2, team 3 plays against team 4 (exactly in this order), and so on (so, 2^{k-1} games are played in that phase). When a team loses a game, it is eliminated, and each game results in elimination of one team (there are no ties). After that, only 2^{k-1} teams remain. If only one team remains, it is declared the champion; otherwise, 2^{k-2} games are played: in the first one of them, the winner of the game "1 vs 2" plays against the winner of the game "3 vs 4", then the winner of the game "5 vs 6" plays against the winner of the game "7 vs 8", and so on. This process repeats until only one team remains.

For example, this picture describes the chronological order of games with $k = 3$:



Let the string s consisting of $2^k - 1$ characters describe the results of the games in chronological order as follows:

- if s_i is `0`, then the team with lower index wins the i-th game;
- if s_i is `1`, then the team with greater index wins the i-th game;
- if s_i is `?`, then the result of the i-th game is unknown (any team could win this game).

Let $f(s)$ be the number of *possible winners* of the tournament described by the string s. A team i is a *possible winner* of the tournament if it is possible to replace every `?` with either `1` or `0` in such a way that team i is the champion.

You are given the initial state of the string s. You have to process q queries of the following form:

- p c — replace s_p with character c, and print $f(s)$ as the result of the query.

Input
The first line contains one integer k ($1 \le k \le 18$).

The second line contains a string consisting of $2^k - 1$ characters — the initial state of the string s. Each character is either `?`, `0`, or `1`.

The third line contains one integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

Then q lines follow, the i-th line contains an integer p and a character c ($1 \le p \le 2^k - 1$; c is either `?`, `0`, or `1`), describing the i-th query.

Output
For each query, print one integer — $f(s)$.

Example

input

```
3
0110?11
6
5 1
6 ?
7 ?
1 ?
5 ?
1 1
```

output

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

E. Gold Transfer

time limit per test: 4.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a rooted tree. Each vertex contains a_i tons of gold, which costs c_i per one ton. Initially, the tree consists only a root numbered 0 with a_0 tons of gold and price c_0 per ton.

There are q queries. Each query has one of two types:

1. Add vertex i (where i is an index of query) as a son to some vertex p_i; vertex i will have a_i tons of gold with c_i per ton. It's guaranteed that $c_i > c_{p_i}$.
2. For a given vertex v_i consider the simple path from v_i to the root. We need to purchase w_i tons of gold from vertices on this path, spending the minimum amount of money. If there isn't enough gold on the path, **we buy all we can**.

If we buy x tons of gold in some vertex v the remaining amount of gold in it decreases by x (of course, we can't buy more gold that vertex has at the moment). For each query of the second type, calculate the resulting amount of gold we bought and the amount of money we should spend.

Note that you should solve the problem in `online` mode. It means that you can't read the whole input at once. You can read each query only after writing the answer for the last query, so don't forget to flush output after printing answers. You can use functions like `fflush(stdout)` in C++ and `BufferedWriter.flush` in Java or similar after each writing in your program. In standard (if you don't tweak I/O), `endl` flushes `cout` in C++ and `System.out.println` in Java (or `println` in Kotlin) makes automatic flush as well.

Input

The first line contains three integers q, a_0 and c_0 ($1 \le q \le 3 \cdot 10^5$; $1 \le a_0, c_0 < 10^6$) — the number of queries, the amount of gold in the root and its price.

Next q lines contain descriptions of queries; The i-th query has one of two types:

- "$1\ p_i\ a_i\ c_i$" ($0 \le p_i < i$; $1 \le a_i, c_i < 10^6$): add vertex i as a son to vertex p_i. The vertex i will have a_i tons of gold with price c_i per one ton. It's guaranteed that p_i exists and $c_i > c_{p_i}$.
- "$2\ v_i\ w_i$" ($0 \le v_i < i$; $1 \le w_i < 10^6$): buy w_i tons of gold from vertices on path from v_i to 0 spending the minimum amount of money. If there isn't enough gold, we buy as much as we can. It's guaranteed that vertex v_i exist.

It's guaranteed that there is at least one query of the second type.

Output

For each query of the second type, print the resulting amount of gold we bought and the minimum amount of money we should spend.

Example

input

```
5 5 2
2 0 2
1 0 3 4
2 2 4
1 0 1 3
2 4 2
```

output

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

Note

Explanation of the sample:

At the first query, the tree consist of root, so we purchase 2 tons of gold and pay $2 \cdot 2 = 4$. 3 tons remain in the root.

At the second query, we add vertex 2 as a son of vertex 0. Vertex 2 now has 3 tons of gold with price 4 per one ton.

At the third query, a path from 2 to 0 consists of only vertices 0 and 2 and since $c_0 < c_2$ we buy 3 remaining tons of gold in vertex 0 and 1 ton in vertex 2. So we bought $3 + 1 = 4$ tons and paid $3 \cdot 2 + 1 \cdot 4 = 10$. Now, in vertex 0 no gold left and 2 tons of gold remain in vertex 2.

At the fourth query, we add vertex 4 as a son of vertex 0. Vertex 4 now has 1 ton of gold with price 3.

At the fifth query, a path from 4 to 0 consists of only vertices 0 and 4. But since no gold left in vertex 0 and only 1 ton is in vertex 4, we buy 1 ton of gold in vertex 4 and spend $1 \cdot 3 = 3$. Now, in vertex 4 no gold left.

F. String Distance

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Suppose you are given two strings a and b. You can apply the following operation any number of times: choose any **contiguous** substring of a or b, and sort the characters in it in non-descending order. Let $f(a, b)$ the minimum number of operations you have to apply in order to make them equal (or $f(a, b) = 1337$ if it is impossible to make a and b equal using these operations).

For example:

- $f(\text{ab}, \text{ab}) = 0$;
- $f(\text{ba}, \text{ab}) = 1$ (in one operation, we can sort the whole first string);
- $f(\text{ebcda}, \text{ecdba}) = 1$ (in one operation, we can sort the substring of the second string starting from the 2-nd character and ending with the 4-th character);
- $f(\text{a}, \text{b}) = 1337$.

You are given n strings $s_1, s_2, \ldots, s_k$ having equal length. Calculate $\sum\limits_{i=1}^{n} \sum\limits_{j=i+1}^{n} f(s_i, s_j)$.

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of strings.

Then n lines follow, each line contains one of the strings s_i, consisting of lowercase Latin letters. $|s_1| = |s_2| = \ldots = |s_n|$, and $n \cdot |s_1| \le 2 \cdot 10^5$. All these strings are pairwise distinct.

Output

Print one integer: $\sum\limits_{i=1}^{n} \sum\limits_{j=i+1}^{n} f(s_i, s_j)$.

Examples

input
4 zzz bac abc acb
output
4015

input
2 a b
output
1337