A. Countdown

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

The "Countdown" TV show has a part that consists of obtaining a number by combining six different numbers using the basic mathematical operations: addition, subtraction, product, and division. The basic rules for the game are:

- The contestant selects six of twenty-four shuffled tiles. The tiles are arranged into two groups: four "large numbers" (25, 50, 75 and 100) and the remainder "small numbers", which comprise two each of the numbers 1 to 10. Hence the tiles have the values {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100}.
- The contestant chooses how many large numbers are in the selection; anywhere from none.
- The contestants then have thirty seconds to get a number as close to the target as possible by combining the six selected numbers using addition, subtraction, multiplication, and division.
- Not all numbers need to be used.
- A number can be used as many times as it appears.
- Fractions are not allowed, only positive integers may be used at any stage of the calculation.

Example

- Contestant requests two large numbers and four small numbers.
- Selection is: $75\ 50\ 2\ 3\ 8\ 7$
- Randomly generated target is: 812
- Contestant declares result: 813
- Contestant gives details: $75 + 50 = 125$; $125 - 8 = 117$; $117 \times 7 = 819$; $3 \times 2 = 6$; $819 - 6 = 813$
- Expert notes: $50 + 8 = 58$; $7 \times 2 = 14$; $14 \times 58 = 812$

Your task is to write a program that calculates the best sequence of operations that lead to the target number T. If there is no way to get T, give the closest solution.

Input

The input consists of several cases, one per line. The first line indicates the number of cases C to process ($1 \leq C \leq 50$). Each of the following C lines contains six natural numbers from the set {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 75, 100} and another natural number T ($1 \leq T \leq 999$) that indicates the target.

Output

The output for each case will be a set of lines with the following format:

- First line: **Target**: number T
- n lines: sequence of operations, the format is $\texttt{operand}_2\ \texttt{operator}\ \texttt{operand}_2 = \texttt{result}$
- Last line: **Best approx**: number obtained
- Blank line

See example for a better understanding. If there is more than one best approximation, all of them will be considered valid. The sequence of operations should be valid, you should never use a value before you obtain it. It is OK to print more operations than needed as long as they are valid. Note that all the numbers and operators must be separated by at least one space

Example

input

```
3
1 75 100 5 3 25 25
100 100 100 100 100 75 345
1 3 1 10 100 75 345
```

output

```
Target: 25
Best approx: 25

Target: 345
100 + 100 = 200
100 + 75 = 175
200 * 175 = 35000
35000 - 100 = 34900
34900 / 100 = 349
Best approx: 349

Target: 345
1 + 3 = 4
100 - 10 = 90
4 - 1 = 3
90 * 3 = 270
75 + 270 = 345
Best approx: 345
```

B. Privatization of Roads in Berland

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

There are n cities and m two-way roads in Berland, each road connecting two distinct cities.

Recently the Berland government has made a tough decision to transfer ownership of the roads to private companies. In total, there are 100500 private companies in Berland, numbered by integers from 1 to 100500. After the privatization, every road should belong to exactly one company.

The anti-monopoly committee demands that after the privatization each company can own at most two roads. The urbanists of Berland also stated their opinion: each city should be adjacent to the roads owned by at most k companies.

Help the government to distribute the roads between the companies so that both conditions are satisfied. That is, each company gets at most two roads, and each city has roads of at most k distinct companies adjacent to it.

Input

Input contains one or several test cases. The first line contains an integer t ($1 \le t \le 300$) — the number of test cases in the input. Solve test cases separately, test cases are completely independent and do not affect each other.

The following lines describe the test cases. Each case starts with a line consisting of three space-separated integers n, m and k ($2 \le n \le 600$, $1 \le m \le 600$, $1 \le k \le n - 1$) — the number of cities, the number of roads and the maximum diversity of the roads adjacent to a city.

Then m lines follow, each having a pair of space-separated integers a_i, b_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$). It means that the i-th road connects cities a_i and b_i. All roads are two-way. There is at most one road between a pair of the cities.

The sum of n values for all test cases doesn't exceed 600. The sum of m values for all test cases doesn't exceed 600.

Output

Print t lines: the i-th line should contain the answer for the i-th test case. For a test case, print a sequence of integers $c_1, c_2, \ldots, c_m$ separated by space, where c_i ($1 \le c_i \le 100500$) is the company which owns the i-th road in your plan. If there are multiple solutions, output any of them. If there is no solution for a test case, print $c_1 = c_2 = \ldots = c_m = 0$.

Example

input

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

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

output

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

C. Draw Brackets!

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

A sequence of square brackets is regular if by inserting symbols "+" and "1" into it, you can get a regular mathematical expression from it. For example, sequences "[[]][]", "[]" and "[[][[]]]" — are regular, at the same time "][", "[[]" and "[[]]][" — are irregular.

Draw the given sequence using a minimalistic pseudographics in the strip of the lowest possible height — use symbols '+', '-' and '|'. For example, the sequence "[[][]][]" should be represented as:

```
+-        -++- -+
|+- -++- -+||   |
||   ||   |||   |
|+- -++- -+||   |
+-        -++- -+
```

Each bracket should be represented with the hepl of one or more symbols '|' (the vertical part) and symbols '+' and '-' as on the example which is given above.

Brackets should be drawn without spaces one by one, only dividing pairs of consecutive pairwise brackets with a single-space bar (so that the two brackets do not visually merge into one symbol). The image should have the minimum possible height.

The enclosed bracket is always smaller than the surrounding bracket, but each bracket separately strives to maximize the height of the image. So the pair of final brackets in the example above occupies the entire height of the image.

Study carefully the examples below, they adequately explain the condition of the problem. Pay attention that in this problem the answer (the image) is unique.

Input

The first line contains an even integer n ($2 \le n \le 100$) — the length of the sequence of brackets.

The second line contains the sequence of brackets — these are n symbols "[" and "]". It is guaranteed that the given sequence of brackets is regular.

Output

Print the drawn bracket sequence in the format which is given in the condition. Don't print extra (unnecessary) spaces.

Examples

input
8 [[][]][]
output
+- -++- -+ |+- -++- -+|| | || || ||| | |+- -++- -+|| | +- -++- -+

input

```
6
[[[]]]
```

output

```
+-     -+
|+-   -+|
||+- -+||
|||   |||
||+- -+||
|+-   -+|
+-     -+
```

input

```
6
[[][]]
```

output

```
+-         -+
|+- -++- -+|
||   ||   ||
|+- -++- -+|
+-         -+
```

input

```
2
[]
```

output

```
+- -+
|   |
+- -+
```

input

```
4
[][]
```

output

```
+- -++- -+
|   ||   |
+- -++- -+
```

D. Pashmak and Graph

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

Pashmak's homework is a problem about graphs. Although he always tries to do his homework completely, he can't solve this problem. As you know, he's really weak at graph theory; so try to help him in solving the problem.

You are given a weighted directed graph with n vertices and m edges. You need to find a path (perhaps, non-simple) with maximum number of edges, such that the weights of the edges increase along the path. In other words, each edge of the path must have strictly greater weight than the previous edge in the path.

Help Pashmak, print the number of edges in the required path.

Input

The first line contains two integers n, m ($2 \le n \le 3 \cdot 10^5$; $1 \le m \le min(n \cdot (n - 1), 3 \cdot 10^5)$). Then, m lines follows. The i-th line contains three space separated integers: u_i, v_i, w_i ($1 \le u_i, v_i \le n$; $1 \le w_i \le 10^5$) which indicates that there's a directed edge with weight w_i from vertex u_i to vertex v_i.

It's guaranteed that the graph doesn't contain self-loops and multiple edges.

Output

Print a single integer — the answer to the problem.

Examples

input
3 3 1 2 1 2 3 1 3 1 1
output
1

input
3 3 1 2 1 2 3 2 3 1 3
output
3

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

```
6
```

Note

In the first sample the maximum trail can be any of this trails: $1 \to 2, 2 \to 3, 3 \to 1$.

In the second sample the maximum trail is $1 \to 2 \to 3 \to 1$.

In the third sample the maximum trail is $1 \to 2 \to 5 \to 4 \to 3 \to 2 \to 6$.

E. Extreme Permutations

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

Permutation of length n is called the sequence of n integers, containing each of integers between 1 and n exactly once. For example, $(3, 4, 5, 1, 2)$ and $(1, 2)$ are permutations, $(1, 4, 3)$ and $(2, 1, 3, 2)$ are not.

Lets call the permutation extreme, if for any two neighbor numbers in the permutation difference between them is not less than minimum of those 2 numbers. For example, permutation $(3, 1, 2, 4)$ is extreme, because $|3 - 1| \geq min(3, 1)$, $|1 - 2| \geq min(1, 2)$ and $|2 - 4| \geq min(2, 4)$.

Given an odd n, calculate number of the extreme permutations of length n where positions of some integers are fixed.

Input

First line of the input contains one integer n — length of permutation ($1 \leq n \leq 27$, $n = 2k + 1$ for some integer k).

Second line contains n integers $p_1, p_2, ..., p_n$ ($0 \leq p_i \leq n$). If p_i is equal to 0, then i-th position is not fixed, otherwise on i-th position must stay p_i. You may assume that if $p_i > 0$ and $p_j > 0$ for $1 \leq i, j \leq n$, $i \neq j$, then $p_i \neq p_j$.

Output

Print one integer — number of the extreme permutations where all fixed elements are on their positions.

Examples

input
5 0 0 0 0 0
output
4

input
5 0 1 0 0 5
output
1

F. Leaf Partition

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

You are given a rooted tree with n nodes, labeled from 1 to n. The tree is rooted at node 1. The parent of the i-th node is p_i. A leaf is node with no children. For a given set of leaves L, let $f(L)$ denote the smallest connected subgraph that contains all leaves L.

You would like to partition the leaves such that for any two different sets x, y of the partition, $f(x)$ and $f(y)$ are disjoint.

Count the number of ways to partition the leaves, modulo 998244353. Two ways are different if there are two leaves such that they are in the same set in one way but in different sets in the other.

Input

The first line contains an integer n ($2 \leq n \leq 200\,000$) — the number of nodes in the tree.

The next line contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \leq p_i < i$).

Output

Print a single integer, the number of ways to partition the leaves, modulo 998244353.

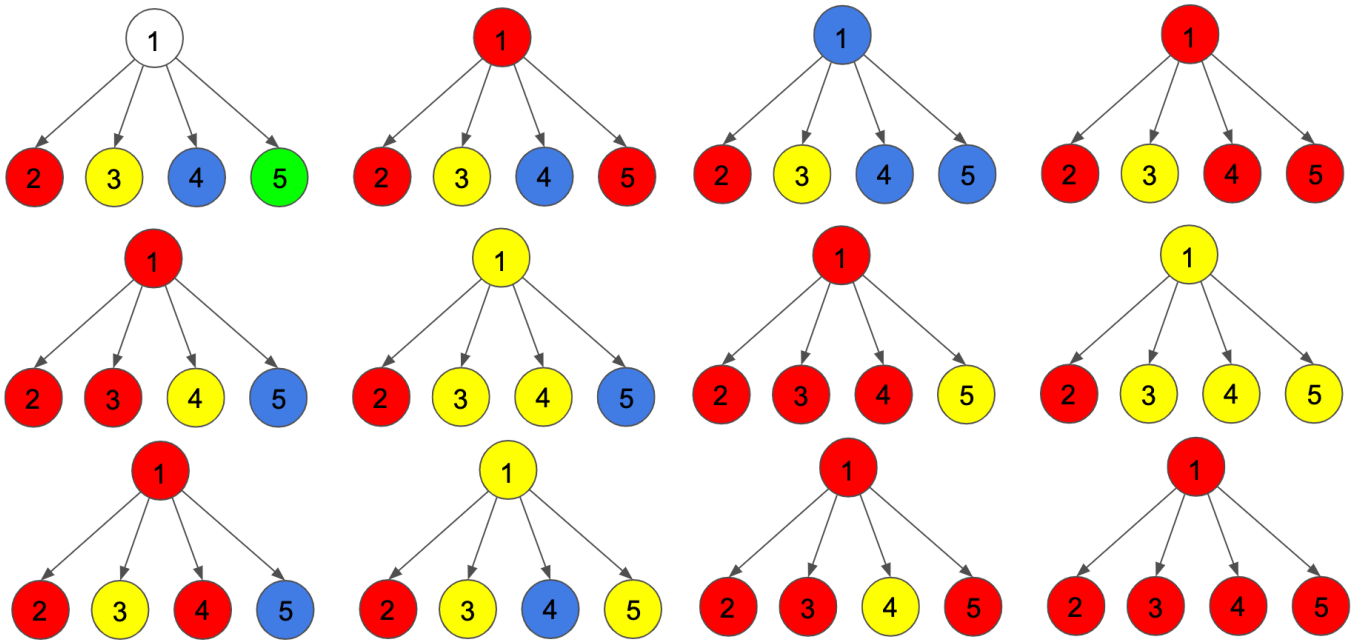
Examples

input
5 1 1 1 1
output
12

input
10 1 2 3 4 5 6 7 8 9
output
1

Note

In the first example, the leaf nodes are $2, 3, 4, 5$. The ways to partition the leaves are in the following image



In the second example, the only leaf is node 10 so there is only one partition. Note that node 1 is not a leaf.

G. Aquarium decoration

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

Arkady and Masha want to choose decorations for thier aquarium in Fishdom game. They have n decorations to choose from, each of them has some cost. To complete a task Arkady and Masha need to choose **exactly** m decorations from given, and they want to spend as little money as possible.

There is one difficulty: Masha likes some a of the given decorations, Arkady likes some b of the given decorations. Some decorations may be liked by both Arkady and Masha, or not be liked by both. The friends want to choose such decorations so that each of them likes **at least** k decorations among the chosen. Help Masha and Arkady find the minimum sum of money they need to spend.

Input

The first line contains three integers n, m and k ($1 \le n \le 200000$, $1 \le m \le n$, $1 \le k \le n$) — the number of decorations, how many decorations the friends should choose, how many decorations each of them should like among the chosen.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$) — decorations costs.

The third line contains single integer a ($1 \le a \le n$) — the number of decorations liked by Masha. The fourth line contains a distinct integers $x_1, x_2, \ldots, x_a$ ($1 \le x_i \le n$) — the ids of decorations liked by Masha.

The next two lines describe decorations liked by Arkady in the same format.

Output

Print single integer: the minimum sum of money the friends should spend to fulfill all constraints. If it is not possible, print `-1`.

Examples

input
4 3 2 3 2 2 1 2 1 2 2 1 3
output
7

input
4 3 2 3 2 2 1 2 1 2 3 4 1 3
output
6

input

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

output

```
-1
```

Note

In the first example the only possible variant to choose 3 decorations having all conditions satisfied is to choose decorations $1, 2, 3$.

In the second example friends can choose decoration 4 instead of decoration 3, because this one is one coin cheaper.

In the third example it's not possible to choose 2 decorations in a way that both are liked by both Masha and Arkady.

H. Virus synthesis

time limit per test: 6 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Viruses are usually bad for your health. How about fighting them with... other viruses? In this problem, you need to find out how to synthesize such good viruses.

We have prepared for you a set of strings of the letters `A`, `G`, `T` and `C`. They correspond to the DNA nucleotide sequences of viruses that we want to synthesize, using the following operations:

- Adding a nucleotide either to the beginning or the end of the existing sequence.
- Replicating the sequence, reversing the copied piece, and gluing it either to the beginning or to the end of the original (so that e.g., `AGTC` can become `AGTCCTGA` or `CTGAAGTC`).

We're concerned about efficiency, since we have very many such sequences, some of them very long. Find a way to synthesize them in a minimum number of operations.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case consists of a single line containing a non-empty string. The string uses only the capital letters `A`, `C`, `G` and `T` and is not longer than 10^5 characters.

Output

For each test case, output a single line containing the minimum total number of operations necessary to construct the given sequence.

Example

input
4 AAAA AGCTTGCA AAGGGGAAGGGGAA AAACAGTCCTGACAAAAAAAAAAAAC

output
3 8 6 18

I. Lucky Pascal Triangle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Do you know what a Pascal Triangle is? No? Need a refresher? Then here are the key points:

- Pascal Triangle is made of multiple rows, starting from 0.
- The i-th row has $i + 1$ elements.
- Each row starts and ends with 1.
- We can consider the Pascal Triangle to be a `2D` array, where `pascal[i][j]` gives us the value of the element in i-th row and j-th column of the Pascal Triangle.
- Except for the first and last column of each row, all other columns can be calculated as `pascal[i][j]` = `pascal[i-1][j]` + `pascal[i-1][j-1]`.

For example, here are the first 10 rows of Pascal Triangle.

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1

Ok. Now that we have revised what a Pascal Triangle is, let us talk about a new kind Pascal Triangle, called "Lucky Pascal Triangle". Legend says that a Lucky Pascal Triangle is a Pascal Triangle containing only elements that are divisible by the ultimate lucky number 7. All other elements that are not divisible by 7 are converted to 0. So, the first 10 rows of a Lucky Pascal Triangle will be:

0
0 0
0 0 0
0 0 0 0
0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0 0
0 7 21 35 35 21 7 0
0 0 28 56 70 56 28 0 0
0 0 0 84 126 126 84 0 0 0

Fascinating right? It seems like there is some kind of pattern in the Lucky Pascal Triangle. We need to investigate more.

As the first step of the investigation, we need to know how many non-zero elements there are in a Lucky Pascal Triangle with N rows. Can you please help us?

Since the answer can be huge, please output your result modulo $10^9 + 7$.

Input

The first line of the input contains a single integer T ($1 \leq T \leq 10^5$) denoting the number of test cases. Next T lines follow with a single non-negative integer N ($N \leq 10^{18}$).

Output

For each value of N, print the test case number and output the number of non-zero elements in the Lucky Pascal Triangle with N rows, modulo $10^9 + 7$. See sample input/output for more details

Example

standard input	standard output
5 1 5 10 15 100000	Case 1: 0 Case 2: 0 Case 3: 18 Case 4: 43 Case 5: 653881477

J. String Problem

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

Boy Valera likes strings. And even more he likes them, when they are identical. That's why in his spare time Valera plays the following game. He takes any two strings, consisting of lower case Latin letters, and tries to make them identical. According to the game rules, with each move Valera can change **one** arbitrary character A_i in one of the strings into arbitrary character B_i, but he has to pay for every move a particular sum of money, equal to W_i. He is allowed to make as many moves as he needs. Since Valera is a very economical boy and never wastes his money, he asked you, an experienced programmer, to help him answer the question: what minimum amount of money should Valera have to get identical strings.

Input

The first input line contains two initial non-empty strings s and t, consisting of lower case Latin letters. The length of each string doesn't exceed 10^5. The following line contains integer n ($0 \le n \le 500$) — amount of possible changings. Then follow n lines, each containing characters A_i and B_i (lower case Latin letters) and integer W_i ($0 \le W_i \le 100$), saying that it's allowed to change character A_i into character B_i in any of the strings and spend sum of money W_i.

Output

If the answer exists, output the answer to the problem, and the resulting string. Otherwise output `-1` in the only line. If the answer is not unique, output any.

Examples

input
uayd uxxd 3 a x 8 x y 13 d c 3

output
21 uxyd

input
a b 3 a b 2 a b 3 b a 5

output
2 b

input
abc ab 6 a b 4

```
a b 7
b a 8
c b 11
c a 3
a c 0
```

output

```
-1
```

K. Lucky Array

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

Petya loves lucky numbers. Everybody knows that lucky numbers are positive integers whose decimal representation contains only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.

Petya has an array consisting of n numbers. He wants to perform m operations of two types:

- add l r d — add an integer d to all elements whose indexes belong to the interval from l to r, inclusive $(1 \le l \le r \le n, 1 \le d \le 10^4)$;
- count l r — find and print on the screen how many lucky numbers there are among elements with indexes that belong to the interval from l to r inclusive $(1 \le l \le r \le n)$. Each lucky number should be counted as many times as it appears in the interval.

Petya has a list of all operations. The operations are such that after all additions the array won't have numbers that would exceed 10^4. Help Petya write a program that would perform these operations.

Input

The first line contains two integers n and m $(1 \le n, m \le 10^5)$ — the number of numbers in the array and the number of operations correspondingly. The second line contains n positive integers, none of which exceeds 10^4 — those are the array numbers. Next m lines contain operations, one per line. They correspond to the description given in the statement.

It is guaranteed that after all operations are fulfilled each number in the array will not exceed 10^4.

Output

For each operation of the second type print the single number on the single line — the number of lucky numbers in the corresponding interval.

Examples

input
3 6 2 3 4 count 1 3 count 1 2 add 1 3 2 count 1 3 add 2 3 3 count 1 3

output
1 0 1 1

input
4 5 4 4 4 4 count 1 4 add 1 4 3

```
count 1 4
add 2 3 40
count 1 4
```

output

```
4
4
4
```

Note

In the first sample after the first addition the array will look in the following manner:

4 5 6

After the second addition:

4 8 9

The second sample after the first addition:

7 7 7 7

After the second addition:

7 47 47 7

L. Small Graph

time limit per test: 1 second

memory limit per test: 256 megabytes

input: small-graph.in

output: small-graph.out

Consider a permutation $p_1, p_2, \ldots, p_n$ of integers between 1 and n. A directed graph G with at least n vertices is called an *inversion graph* of that permutation when for any i, j such that $1 \le i, j \le n$, $i \ne j$ the j-th vertex of G is reachable via some path from the i-th vertex of G *if and only if* $i < j$ and $p_i > p_j$ (such pair is called an *inversion* of the permutation).

Naturally, there exist many inversion graphs for each permutation. You need to find a relatively small one: the graph must contain at most $30n$ vertices and at most $30n$ edges.

Input

The first line of the input file contains one integer n, $1 \le n \le 1000$, denoting the size of the permutation.

The second line of the input file contains the permutation itself.

ATTENTION: This problem doesn't have standard input/output.

Output

In the first line of the output file, print two integers v and a — the number of vertices and arcs of the graph, respectively. The next a lines should contain the arcs. Each arc should be described by two vertex numbers (between 1 and v) — the source and destination of the arc.

The number v should be at least n and at most $30n$, a should be at least 0 and at most $30n$.

ATTENTION: This problem doesn't have standard input/output.

Example

input
4 3 4 1 2
output
5 4 1 5 2 5 5 3 5 4