Educational Codeforces Round 113 (Rated for Div. 2)

A. Balanced Substring

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 n letters, each letter is either 'a' or 'b'. The letters in the string are numbered from 1 to n.

$s[l; r]$ is a continuous substring of letters from index l to r of the string inclusive.

A string is called balanced if the number of letters 'a' in it is equal to the number of letters 'b'. For example, strings "baba" and "aabbab" are balanced and strings "aaab" and "b" are not.

Find any non-empty balanced substring $s[l; r]$ of string s. Print its l and r ($1 \le l \le r \le n$). If there is no such substring, then print -1 -1.

Input

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

Then the descriptions of t testcases follow.

The first line of the testcase contains a single integer n ($1 \le n \le 50$) — the length of the string.

The second line of the testcase contains a string s, consisting of n letters, each letter is either 'a' or 'b'.

Output

For each testcase print two integers. If there exists a non-empty balanced substring $s[l; r]$, then print l r ($1 \le l \le r \le n$). Otherwise, print -1 -1.

Example

input
4 1 a 6 abbaba 6 abbaba 9 babbabbaa
output
-1 -1 1 6 3 6 2 5

Note

In the first testcase there are no non-empty balanced subtrings.

In the second and third testcases there are multiple balanced substrings, including the entire string "abbaba" and substring "baba".

B. Chess Tournament

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

A chess tournament will be held soon, where n chess players will take part. Every participant will play one game against every other participant. Each game ends in either a win for one player and a loss for another player, or a draw for both players.

Each of the players has their own expectations about the tournament, they can be one of two types:

1. a player wants not to lose any game (i. e. finish the tournament with **zero losses**);
2. a player wants to win at least one game.

You have to determine if there exists an outcome for all the matches such that all the players meet their expectations. If there are several possible outcomes, print any of them. If there are none, report that it's impossible.

Input

The first line contains a single integer t ($1 \le t \le 200$) — the number of test cases.

The first line of each test case contains one integer n ($2 \le n \le 50$) — the number of chess players.

The second line contains the string s ($|s| = n$; $s_i \in \{1, 2\}$). If $s_i = 1$, then the i-th player has expectations of the first type, otherwise of the second type.

Output

For each test case, print the answer in the following format:

In the first line, print `NO` if it is impossible to meet the expectations of all players.

Otherwise, print YES, and the matrix of size $n \times n$ in the next n lines.

The matrix element in the i-th row and j-th column should be equal to:

- +, if the i-th player won in a game against the j-th player;
- -, if the i-th player lost in a game against the j-th player;
- =, if the i-th and j-th players' game resulted in a draw;
- X, if $i = j$.

Example

input
3 3 111 2 21 4 2122
output
YES X== =X= ==X NO YES X--+ +X++ +-X- --+X

C. Jury Meeting

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

n people gathered to hold a jury meeting of the upcoming competition, the i-th member of the jury came up with a_i tasks, which they want to share with each other.

First, the jury decides on the order which they will follow while describing the tasks. Let that be a permutation p of numbers from 1 to n (an array of size n where each integer from 1 to n occurs exactly once).

Then the discussion goes as follows:

- If a jury member p_1 has some tasks left to tell, then they tell one task to others. Otherwise, they are skipped.
- If a jury member p_2 has some tasks left to tell, then they tell one task to others. Otherwise, they are skipped.
- ...
- If a jury member p_n has some tasks left to tell, then they tell one task to others. Otherwise, they are skipped.
- If there are still members with tasks left, then the process repeats from the start. Otherwise, the discussion ends.

A permutation p is nice if none of the jury members tell two or more of their own tasks in a row.

Count the number of nice permutations. The answer may be really large, so print it modulo $998\,244\,353$.

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

The first line of the test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — number of jury members.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the number of problems that the i-th member of the jury came up with.

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case, print one integer — the number of nice permutations, taken modulo $998\,244\,353$.

Example

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

output
1 6 0 540

Note
Explanation of the first test case from the example:

There are two possible permutations, $p = [1, 2]$ and $p = [2, 1]$. For $p = [1, 2]$, the process is the following:

1. the first jury member tells a task;
2. the second jury member tells a task;
3. the first jury member doesn't have any tasks left to tell, so they are skipped;
4. the second jury member tells a task.

So, the second jury member has told two tasks in a row (in succession), so the permutation is not nice.

For $p = [2, 1]$, the process is the following:

1. the second jury member tells a task;
2. the first jury member tells a task;
3. the second jury member tells a task.

So, this permutation is nice.

D. Inconvenient Pairs

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

There is a city that can be represented as a square grid with corner points in $(0, 0)$ and $(10^6, 10^6)$.

The city has n vertical and m horizontal streets that goes across the whole city, i. e. the i-th vertical streets goes from $(x_i, 0)$ to $(x_i, 10^6)$ and the j-th horizontal street goes from $(0, y_j)$ to $(10^6, y_j)$.

All streets are bidirectional. Borders of the city are streets as well.

There are k persons staying *on the streets*: the p-th person at point (x_p, y_p) (so either x_p equal to some x_i or y_p equal to some y_j, or both).

Let's say that a pair of persons form an *inconvenient pair* if the shortest path from one person to another going only by streets is *strictly greater* than the Manhattan distance between them.

Calculate the number of inconvenient pairs of persons (pairs (x, y) and (y, x) are the same pair).

Let's recall that Manhattan distance between points (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$.

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

The first line of each test case contains three integers n, m and k ($2 \le n, m \le 2 \cdot 10^5$; $2 \le k \le 3 \cdot 10^5$) — the number of vertical and horizontal streets and the number of persons.

The second line of each test case contains n integers $x_1, x_2, \ldots, x_n$ ($0 = x_1 < x_2 < \cdots < x_{n-1} < x_n = 10^6$) — the x-coordinates of vertical streets.

The third line contains m integers $y_1, y_2, \ldots, y_m$ ($0 = y_1 < y_2 < \cdots < y_{m-1} < y_m = 10^6$) — the y-coordinates of horizontal streets.

Next k lines contains description of people. The p-th line contains two integers x_p and y_p ($0 \le x_p, y_p \le 10^6$; $x_p \in \{x_1, \ldots, x_n\}$ or $y_p \in \{y_1, \ldots, y_m\}$) — the coordinates of the p-th person. All points are distinct.

It guaranteed that sum of n doesn't exceed $2 \cdot 10^5$, sum of m doesn't exceed $2 \cdot 10^5$ and sum of k doesn't exceed $3 \cdot 10^5$.

Output
For each test case, print the number of inconvenient pairs.

Example

input
2

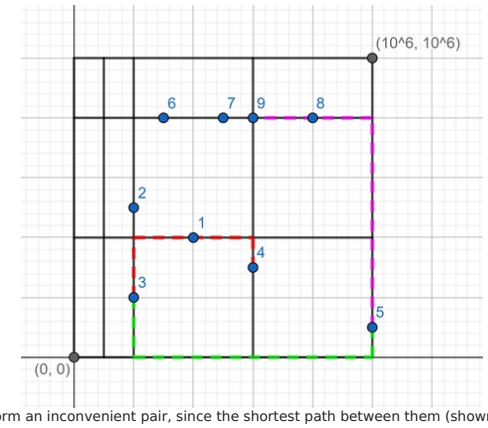
```
2 2 4
0 1000000
0 1000000
1 0
1000000 1
999999 1000000
0 999999
5 4 9
0 1 2 6 1000000
0 4 8 1000000
4 4
2 5
2 2
6 3
1000000 1
3 8
5 8
8 8
6 8
```

output

```
2
5
```

Note

The second test case is pictured below:



For example, points 3 and 4 form an inconvenient pair, since the shortest path between them (shown red and equal to 7) is greater than its Manhattan distance (equal to 5).

Points 3 and 5 also form an inconvenient pair: the shortest path equal to 1000001 (shown green) is greater than the Manhattan distance equal to 999999.

But points 5 and 9 don't form an inconvenient pair, since the shortest path (shown purple) is equal to its Manhattan distance.

E. Playoff Restoration

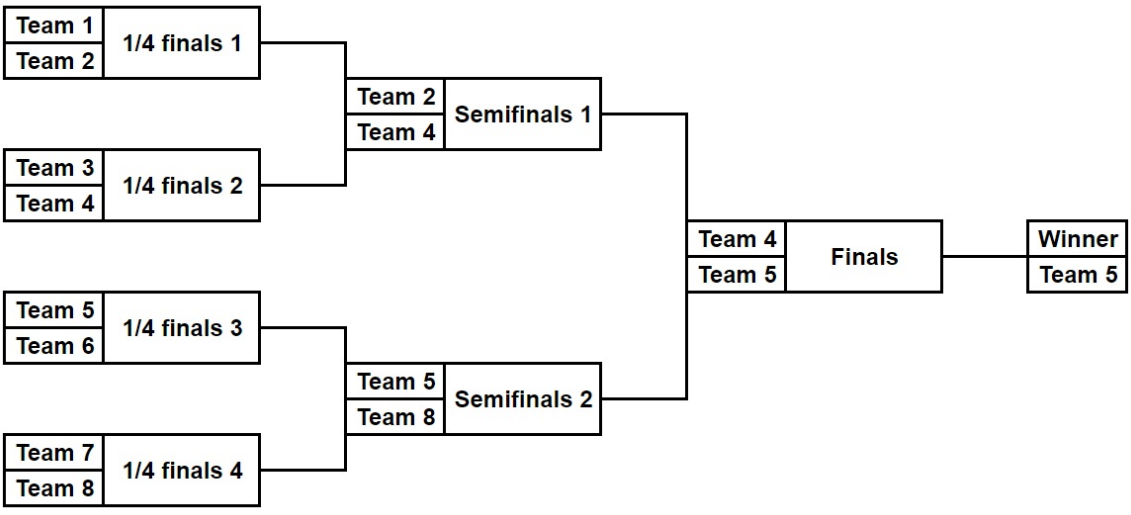
time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

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.

After the tournament ends, the teams are assigned places according to the tournament phase when they were eliminated. In particular:

- the winner of the tournament gets place 1;
- the team eliminated in the finals gets place 2;
- both teams eliminated in the semifinals get place 3;
- all teams eliminated in the quarterfinals get place 5;
- all teams eliminated in the 1/8 finals get place 9, and so on.

For example, this picture describes one of the possible ways the tournament can go with $k = 3$, and the resulting places of the teams:



After a tournament which was conducted by the aforementioned rules ended, its results were encoded in the following way. Let p_i be the place of the i-th team in the tournament. The hash value of the tournament h is calculated as

$h = (\sum_{i=1}^{2^k} i \cdot A^{p_i}) \bmod 998244353$, where A is some given integer.

Unfortunately, due to a system crash, almost all tournament-related data was lost. The only pieces of data that remain are the values of k, A and h. You are asked to restore the resulting placing of the teams in the tournament, if it is possible at all.

Input
The only line contains three integers k, A and h ($1 \le k \le 5$; $100 \le A \le 10^8$; $0 \le h \le 998244352$).

Output
If it is impossible to find any placing of the teams that is consistent with the data you have, print one integer -1.

Otherwise, print 2^k integers, where i-th integer should be equal to p_i (the place of the i-th team). Note that your answer should be consistent with one of the possible ways the tournament could go, and note that the initial structure of the tournament is fixed (for example, teams 1 and 2 always play in the first phase of the tournament against each other). If there are multiple ways to restore the places of the teams which are consistent with the data you have, print any of them.

Examples

input
3 1337 75275197
output
5 3 5 2 1 5 5 3

input
2 100 5040100
output
1 3 3 2

input
2 100 7020100
output
-1

Note
The tournament for the first example is described in the statement.

For the third example, the placing $[1, 2, 3, 3]$ (team 1 gets place 1, team 2 gets place 2, teams 3 and 4 get place 3) could result in a hash value of 7020100 with $A = 100$, but no tournament can result in such placing since teams 1 and 2 play against each other in the semifinals, so they cannot get two first places.

F. Palindromic Hamiltonian Path

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

You are given a simple undirected graph with n vertices, n is even. You are going to write a letter on each vertex. Each letter should be one of the first k letters of the Latin alphabet.

A path in the graph is called Hamiltonian if it visits each vertex exactly once. A string is called palindromic if it reads the same from left to right and from right to left. A path in the graph is called palindromic if the letters on the vertices in it spell a palindromic string without changing the order.

A string of length n is good if:

- each letter is one of the first k lowercase Latin letters;
- if you write the i-th letter of the string on the i-th vertex of the graph, there will exist a palindromic Hamiltonian path in the graph.

Note that the path doesn't necessarily go through the vertices in order $1, 2, \ldots, n$.

Count the number of good strings.

Input
The first line contains three integers n, m and k ($2 \le n \le 12$; n is even; $0 \le m \le \frac{n \cdot (n-1)}{2}$; $1 \le k \le 12$) — the number of vertices in the graph, the number of edges in the graph and the number of first letters of the Latin alphabet that can be used.

Each of the next m lines contains two integers v and u ($1 \le v, u \le n$; $v \ne u$) — the edges of the graph. The graph doesn't contain multiple edges and self-loops.

Output
Print a single integer — number of good strings.

Examples

input
4 3 3 1 2 2 3 3 4
output
9

input
4 6 3 1 2 1 3 1 4 2 3 2 4 3 4
output
21

input
12 19 12 1 3 2 6 3 6 3 7 4 8 8 5 8 7 9 4 5 9 10 1 10 4 10 6 9 10 11 1 5 11 7 11 12 2 12 5 12 11
output
456165084