Problem A: Assembling Triangles

Bash has n segments. The i-th segment has length 2^{a_i}. Bash wants to select three different segments to form a triangle.

Now Bash wonders, how many different ways there are to choose three segments that form a triangle? Note that three segments can form a triangle if and only if the sum of any two segments is greater than the third segment.

Input

The first line contains a single integer t $(1 \le t \le 10^5)$ – the number of test cases. t test cases follow, each consists of two lines:

- The first line contains an integer n $(3 \le n \le 3 \cdot 10^5)$.
- The second line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^9)$.

The sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print a single line containing the number of ways to choose three segments that can form a triangle.

Sample Explanation

In the first test case, there are 3 segments with length 2, 4 and 8. You cannot form any triangle with these three segments.

In the second test case, there are 4 segments, all with length 2. Using any three of these segments, you can form a triangle.

Sample Input 1	Sample Output 1

2
3
1 2 3
4
1 1 1 1
	0
	4

Problem B: Boulevard Blueprint

A newly planned urban area in Hanoi city has a beautiful layout. The urban area is in the shape of a rectangle with dimensions $w \cdot h$. The bottom-left corner of the urban area is at point $(0, 0)$, and the top-right corner is at point (w, h). This urban area has $w + 1$ vertical boulevards that run evenly spaced at $x = 0, x = 1, x = 2, \ldots, x = w$. It also has $h + 1$ horizontal boulevards that run evenly spaced at $y = 0, y = 1, y = 2, \ldots, y = h$.

The management of the urban area wants to assign directions to nearly all boulevards, allowing **at most one** to remain two-way to help reduce congestion. Vertical boulevards should either go from North to South or South to North, and horizontal boulevards should either go from West to East or East to West.

There are n prioritized movement requests between two locations (u_j, v_j) and (p_j, q_j) $(1 \leq j \leq n)$. A prioritized movement request between two locations (u_j, v_j) and (p_j, q_j) is considered satisfied if **both conditions** below are met:

- It is possible to move from (u_j, v_j) to (p_j, q_j) with **at most one *turn***.
- It is possible to move from (p_j, q_j) to (u_j, v_j) with **at most one *turn***.

A *turn* is a change in travel direction from a vertical boulevard to a horizontal boulevard, or from a horizontal boulevard to a vertical boulevard.

You need to count the number of ways to set the directions of the boulevards to satisfy all the movement requests, while keeping **at most one** boulevard two-way. Two configurations are considered different if there exists at least one boulevard that is assigned a different direction. As the result may be large, you only need to find its remainder after dividing $998\,244\,353$.

Input

The first line contains a single positive integer t $(t \leq 10^5)$ representing the number of test cases. The description of the test cases follows.

The first line contains three positive integers w, h and n $(1 \leq w, h \leq 2 \cdot 10^5, 1 \leq n \leq 10^5)$ – the width of the urban area, the height of the urban area, and the number of priority requests, respectively.

The j-th line of the next n lines contains four integers u_j, v_j, p_j, q_j $(0 \leq u_j, p_j \leq w, 0 \leq v_j, q_j \leq h)$ – the j-th prioritized movement request.

It is guaranteed that:

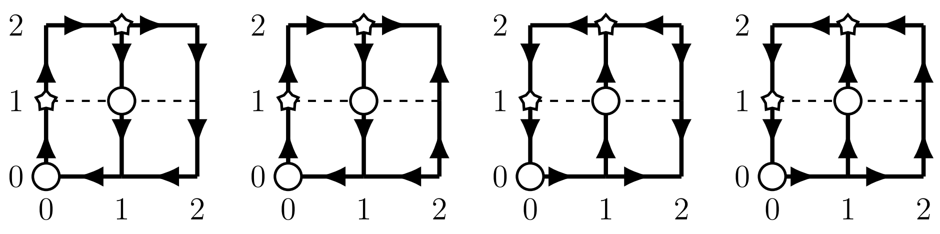
- the sum of w over all test cases does not exceed $2 \cdot 10^5$,
- the sum of h over all test cases does not exceed $2 \cdot 10^5$, and
- the sum of n over all test cases does not exceed 10^5.

Output

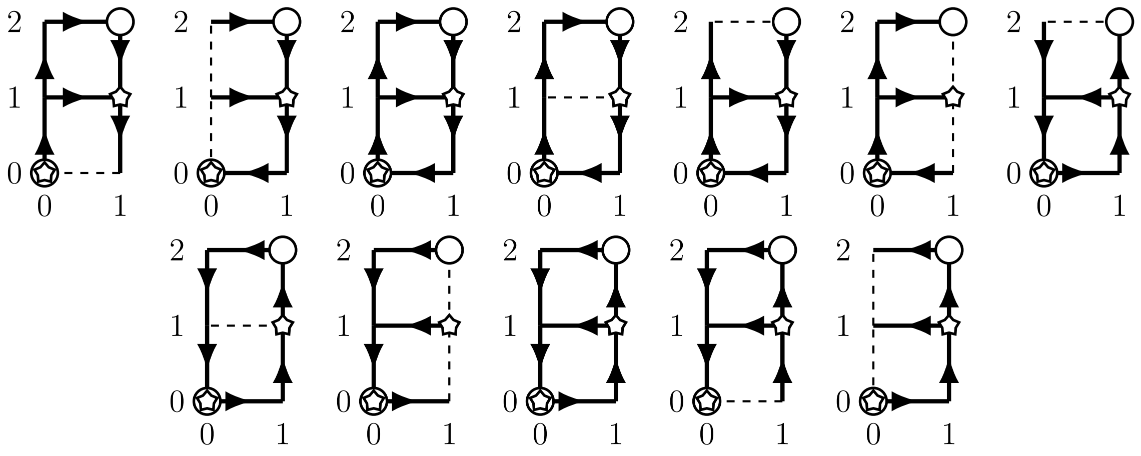
For each test case, print the number of ways to set the direction of boulevards. As the result may be large, you only need to print its remainder after dividing $998\,244\,353$.

Sample Explanation

Below are illustrations for all possible configurations of the test cases. The dashed lines represent the **two-way** boulevards. Locations with the same shape represent the pair of locations with prioritized movement requests.



All possible configurations in the first test case



All possible configurations in the second test case

Sample Input 1

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

Sample Output 1

```
4
12
14
0
```

Problem C: Consecutive Card Game

GSPVH is attending a card game event. He is going to play the game called "Cards in a black box". If he wins the game, he will get a huge prize like a jackpot!

The game goes as follow. At the beginning, the host presents a black box with no cards inside. Then, he puts into the black box n cards. Each card has an integer written on it. The values written on these n cards are $v_1, v_2, \ldots, v_n$. After that, the host annouces a positive integer k.

Now, GSPVH plays a *warmup* round:

- GSPVH tells the host a positive integer t_0.
- The host randomly takes t_0 cards out of the black box.
- If among these cards, there are k cards whose numbers are k consecutive integers, GSPVH wins this round. In other words, GSPVH wins this round if there exists an integer x such that, for every integer y between x and $x + k - 1$ (inclusive), a card with value y is taken out. Otherwise, GSPVH loses this round.
- If GSPVH loses, the game ends immediately and GSPVH will not get any prizes. If GSPVH wins, all chosen cards are put back into the black box, and the game continues.

If the game is still on, GSPVH will play q more rounds consecutively. These rounds are numbered from 1 to q. The i-th round goes as below:

- The host annouces three integers l_i, r_i and c_i.
- For every integer j between l_i and r_i (inclusive), the host first removes **all** cards with value j inside the black box and **burns them forever**, then puts into the black box **exactly** c_i new cards with value j. In other words, after this step, for every integer j between l_i and r_i (inclusive), there are **exactly** c_i cards with value j.
- GSPVH tells the host a positive integer t_i.
- The host randomly takes t_i cards out of the black box.
- If among these cards, there are k cards whose numbers are k consecutive integers, GSPVH wins this round. Otherwise, GSPVH loses this round.
- If GSPVH loses, the game ends immediately and GSPVH will not get any prizes. If GSPVH wins, all chosen cards are put back into the black box, and the game continues.

GSPVH will get the prize only if GSPVH wins all these q rounds.

Please note that:

- In every round, the integer GSPVH tells must be less than or equal to the number of cards inside the black box.
- In every round, GSPVH decides the number of cards which will be taken out, but GSPVH can not control which exact cards will be taken out.
- The value k stays the same throughout the game.
- In every round, if GSPVH wins, all cards which are taken out will be put back into the black box before the next round. Hence, in every round, the set of cards inside the black box does not depend on which cards are taken out in previous rounds.

As the prize is huge, GSPVH wants a sure win. More precisely, in every round, GSPVH wants to choose the number of cards to be taken out so that, no matter which cards the host takes out, GSPVH wins for sure. Also, GSPVH wants the number he tells to be as small as possible.

Help GSPVH decide, for each round, the minimum number of cards to be taken out, so that GSPVH will surely get the prize!

Input

The first line contains three integers n, k and q $(1 \le n \le 10^6, 1 \le q \le 10^5, 1 \le k \le 5)$.

The second line contains n integers $v_1, v_2, \ldots, v_n$ $(-10^9 \le v_i \le 10^9)$.

In the last q lines, the i-th one contains three integers l_i, r_i and c_i $(-10^9 \le l_i \le r_i \le 10^9, 1 \le c_i \le 10^9)$.

It is guaranteed that, in every round, before GSPVH tells a number, the black box contains cards whose numbers are k consecutive integers.

Output

Print $q + 1$ integers, which are the minimum number of cards to ensure GSPVH's win in the *warmup* round, and q next rounds, respectively. Each integer is printed in a separate line.

Sample Explanation

In every turn, GSPVH must make sure that among cards which are taken out, there are $k = 2$ cards with consecutive integers.

In the *warmup* round, the black box contains 4 cards with values 1, 2, 4, 5. If 3 cards are taken out, among them, there must be either both 1 and 2 or both 4 and 5. Hence, GSPVH wins for sure.

In the first round, the black box contains 5 cards with values 1, 2, 4, 5, 8. In this case, the minimum number of cards to guarantee a win is 4.

In the second round, the black box contains 7 cards with values 1, 2, 4, 5, 6, 7, 8. Taking out 4 cards does not guarantee a win, as there might be the case (1, 4, 6, 8). However, taking out 5 cards ensures a win, there are three possible cases:

- Both 1 and 2 are taken out.
- Exactly one of (1, 2) and 4 of (4, 5, 6, 7, 8) are taken out.
- All cards of (4, 5, 6, 7, 8) are taken out.

Sample Input 1

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

Sample Output 1

```
3 4 5
```

Problem D: Dune

On the desert planet *Dune*, nomadic tribes navigate the treacherous sands using rhythmic signals from devices known as **Desert Drums**. These drums emit sounds in a specific pattern to guide travelers safely through the shifting landscape. Each Desert Drum operates in a continuous cycle:

- **Guiding Beat** for exactly G minutes,
- **Warning Beat** for exactly W minutes,
- **Resting Phase** for exactly R minutes.

This sequence repeats indefinitely.

T and a half minutes have passed since a Desert Drum was activated, determine the drum's current phase.

Input

The first line of the input contains a single integer n $(1 \le n \le 1000)$ – the number of test cases.

Each test case consists of a single line, containing integers G, W, R, and T $(1 \le G, W, R \le 1000, 0 \le T \le 3000)$ – the durations (in minutes) of the Guiding Beat, Warning Beat, and Resting Phase, respectively, and T indicating that T and a half minutes have passed since the drum was activated.

Output

For each test case, output a single line containing one of the following strings: `Guiding Beat`, `Warning Beat` or `Resting Phase`, indicating the drum's current phase.

Sample Explanation

In the first test case:

- Between $[0, 3)$ minutes, it is Guiding Beat,
- Between $[3, 8)$ minutes, it is Warning Beat,
- Between $[8, 10)$ minutes, it is Resting Phase,
- Between $[10, 13)$ minutes, it is Guiding Beat,
- Between $[13, 18)$ minutes, it is Warning Beat,
- and so on...

Sample Input 1

```
2
3 5 2 6
4 4 4 11
```

Sample Output 1

```
Warning Beat
Resting Phase
```

This page is intentionally left blank.

Problem E: Edge Elimination

You are given a simple, connected, undirected, **planar** graph, consisting of n vertices and m edges. A graph is planar when it can be drawn on the plane in such a way that its edges intersect only at their endpoints.

Count the number of ways to select **exactly 3 different** edges to remove from the graph, such that the resulting graph is no longer connected.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 3 \cdot 10^5$). The description of the test cases follows.

The first line contains two integers n and m ($3 \le n, m \le 3 \cdot 10^5$) – the number of vertices and the number of edges of the graph, respectively.

The following n lines, the i-th of which contains two integers x_i and y_i ($|x_i|, |y_i| \le 10^9$) – the coordinates of the i-th point.

The following m lines each contain two integers u and v ($1 \le u, v \le n$, $u \ne v$) denoting an edge connecting vertices u and v.

It is guaranteed that:

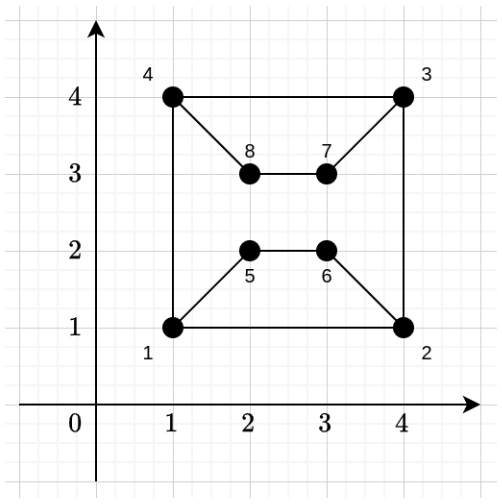
- the graph is connected,
- no two points have the same coordinates,
- no two edges intersect each other except at their common endpoints (if any),
- for each pair of the given points, there is at most one edge connecting them,
- the sum of n over all test cases does not exceed $3 \cdot 10^5$,
- the sum of m over all test cases does not exceed $3 \cdot 10^5$.

Output

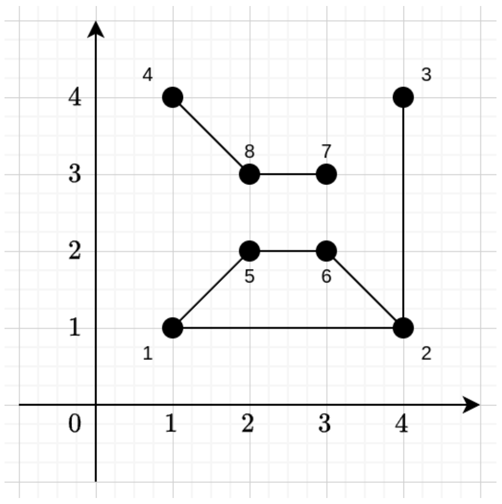
For each test case, output a single integer denoting the number of ways to select exactly 3 edges to disconnect the graph.

Sample Explanation

The following images depict the graph given in the sample test case:



The original graph



One possible way to disconnect the graph: select edges 3, 4, and 9.

Sample Input 1

```
1
8 10
1 1
4 1
4 4
1 4
2 2
3 2
3 3
2 3
1 2
2 3
3 4
4 1
5 6
7 8
1 5
2 6
3 7
4 8
```

Sample Output 1

```
64
```

Problem F: Flurried Alice

Upon learning a new powerful skill called "Rage Rush", Alice immediately heads to her training ground to get accustomed to the ability.

The behavior of this skill is pretty unique. Assuming Alice was using this skill when there are k alive enemy Marchens, and their health points are $b_1, b_2, \ldots, b_k$ respectively (an enemy is considered "alive" if it has at least 1 health point), the following happens:

- First, one Marchen is chosen out of the k alive ones above with a probability proportional to their current health. In other words, denoting p_i as the probability that the i-th Marchen is chosen, then $p_i = \frac{b_i}{\sum_{j=1}^{k} b_j}$.
- Then, an integer in the range $[3, 5]$ is chosen, that each integer in the range has an equal chance of being chosen, and finally decreases the health points of the chosen Marchen by that chosen integer. If that Marchen happened to have fewer health points than the subtracting value, the exceeded damage will be considered wasted (i.e., not being transferred to any other Marchens).

Within the training ground, Alice has set up n dummy Marchens, with their health points being at $a_1, a_2, \ldots, a_n$ respectively, and they don't fight back so that she could train for as long as she likes.

After a while, Gretel comes over; and upon witnessing Alice's flurry upon the dummies, she gives a riddle: if Alice can only use Rage Rush as an attacking option, what is the expected amount of times she uses that skill such that all dummy Marchens will be dead?

"There is so much randomness here, how would you even tell?" — Alice turns to Gretel with a raised brow.

"Hehe... it is actually very simple." — Gretel only gives out her signature wide smile, then utters the answer right afterwards.

Can you calculate it as fast as Gretel? Find the answer modulo 998 224 353.

Formally, let $M = 998\,244\,353$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \bmod M$. In other words, output such an integer x that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Input

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

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the number of dummy Marchens.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{18}$) — the number of health points each dummy Marchen has initially.

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

Output

For each test case, output a single integer — the expected number of Rage Rush uses from Alice to clear all her dummy Marchens, modulo $998\,244\,353$.

Sample Explanation

Denote "a takes b" as a turn with the a-th Marchen taking b damage.

Denote "a to zero" as a turn where the a-th Marchen will be dead after taking damage.

In the first test case, the initial Marchens has health points represented by the pair $(3, 5)$. There are a few scenarios of clearing all Marchens:

Scenario	**Chance of happening**
$(\underline{3}, 5) \xrightarrow{\text{1 to zero}} (0, \underline{5}) \xrightarrow{\text{2 takes 3}} (0, \underline{2}) \xrightarrow{\text{2 to zero}} (0, 0)$	This has $\frac{3}{3+5} \cdot \frac{1}{3} = \frac{1}{8}$ chance. • "1 to zero" has $\frac{3}{3+5}$ chance. • "2 takes 3" has $\frac{1}{3}$ chance.
$(\underline{3}, 5) \xrightarrow{\text{1 to zero}} (0, \underline{5}) \xrightarrow{\text{2 takes 4}} (0, \underline{1}) \xrightarrow{\text{2 to zero}} (0, 0)$	$\frac{3}{3+5} \cdot \frac{1}{3} = \frac{1}{8}$
$(\underline{3}, 5) \xrightarrow{\text{1 to zero}} (0, \underline{5}) \xrightarrow{\text{2 takes 5}} (0, 0)$	$\frac{3}{3+5} \cdot \frac{1}{3} = \frac{1}{8}$
$(3, \underline{5}) \xrightarrow{\text{2 takes 3}} (\underline{3}, 2) \xrightarrow{\text{1 to zero}} (0, \underline{2}) \xrightarrow{\text{2 to zero}} (0, 0)$	$\frac{5}{(3+5) \cdot 3} \cdot \frac{3}{3+2} = \frac{1}{8}$
$(3, \underline{5}) \xrightarrow{\text{2 takes 3}} (3, \underline{2}) \xrightarrow{\text{2 to zero}} (\underline{3}, 0) \xrightarrow{\text{1 to zero}} (0, 0)$	$\frac{5}{(3+5) \cdot 3} \cdot \frac{2}{3+2} = \frac{1}{12}$
$(3, \underline{5}) \xrightarrow{\text{2 takes 4}} (\underline{3}, 1) \xrightarrow{\text{1 to zero}} (0, \underline{2}) \xrightarrow{\text{2 to zero}} (0, 0)$	$\frac{5}{(3+5) \cdot 3} \cdot \frac{3}{3+1} = \frac{5}{32}$
$(3, \underline{5}) \xrightarrow{\text{2 takes 4}} (3, \underline{1}) \xrightarrow{\text{2 to zero}} (\underline{3}, 0) \xrightarrow{\text{1 to zero}} (0, 0)$	$\frac{5}{(3+5) \cdot 3} \cdot \frac{1}{3+1} = \frac{5}{96}$
$(3, \underline{5}) \xrightarrow{\text{2 takes 5}} (\underline{3}, 0) \xrightarrow{\text{1 to zero}} (0, 0)$	$\frac{5}{(3+5) \cdot 3} = \frac{5}{24}$

In total, the expected amount of times Alice will uses the skill is:

$$3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} + 2 \cdot \frac{1}{8} + 3 \cdot \frac{1}{8} + 3 \cdot \frac{1}{12} + 3 \cdot \frac{5}{32} + 3 \cdot \frac{5}{96} + 2 \cdot \frac{5}{24} = \frac{8}{3}$$

It can be seen that $665\,496\,238 \cdot 3 = 1\,996\,488\,714 = 2 \cdot 998\,244\,353 + 8$, or $665\,496\,238 \cdot 3 \equiv 8 \pmod{998\,244\,353}$, thus we output $665\,496\,238$ as the answer for this test case.

Sample Input 1

```
3
2
3 5
3
3 4 5
5
1 4 2 9 5
```

Sample Output 1

```
665496238
4
665496243
```

Problem G: Greedy Grouping

Given an array A with n elements $a_1, a_2, \ldots, a_n$ and an integer k, select a set of segments satisfying the following conditions:

- Each segment consists of consecutive elements of A,
- Each segment has at least k elements,
- The elements within each segment are pairwise distinct,
- No two segments share any elements, i.e. each element of A appears in at most one segment.

Your task is to determine the maximum number of such segments, and compute the number of ways to select the segments to achieve that maximum number.

Two ways are considered different if either of the following conditions hold:

- There exists an element a_i which belongs to some segment in one way, but does not belong to any segments in the other way.
- There exists two elements a_i and a_j which belong to the same segment in one way, but belong to different segments in the other way.

To make the problem a little bit more challenging, you are also given q integers $l_1, l_2, \ldots, l_q$. For each value l_i, you need to solve the above problem where $k = l_i$.

Input

The first line of the input contains an integer t, the number of test cases.

Each test case consists of three lines:

- The first line contains two integers n and q $(1 \leq n \leq 666\,666, 1 \leq q \leq 333)$.
- The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 999\,999\,999)$.
- The third line contains q integers $l_1, l_2, \ldots, l_q$ $(1 \leq l_i \leq n)$.

It is guaranteed that:

- The sum of n over all testcases does not exceed $666\,666$.
- The sum of q over all testcases does not exceed $666\,666$.

Output

For each test case, output q lines. The i-th one among them contains results of the above problem where $k = l_i$, which are represented by two space-separated integers:

- The first one is the maximum number of segments that can be selected.
- The second one is the number of ways to select segments to achieve that maximum number, modulo $998\,244\,353$.

Sample Explanation

Denote $[l, r]$ the segment containing all elements $a_l, a_{l+1}, \ldots, a_r$.

In the first test case:

- When $k = 2$, we can select at most 2 segments. There are 3 valid solutions:
  1. $[1, 2]$ and $[3, 4]$.
  2. $[1, 2]$ and $[4, 5]$.
  3. $[2, 3]$ and $[4, 5]$.
- When $k = 3$, no segments of length at least k contains pairwise distinct elements.

In the second test case:

- When $k = 3$, we can select at most 2 segments. The only valid solution contains two semgents $[1, 3]$ and $[4, 6]$.
- When $k = 5$, we can select only one segment. It can be any of the following: $[1, 5]$ $[2, 6]$, and $[1, 6]$.

Sample Input 1

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

Sample Output 1

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

Problem H: Hardcoded Median

For any array a of n non-negative integers $a_0, a_1, \ldots, a_{n-1}$, where n is **odd**, the median of a is defined as the $\frac{n-1}{2}$-th element of a when the elements are sorted in increasing order. For example:

- For $a = [5, 7, 8]$, the median of a is 7 (as a is already sorted, and $a_{\frac{n-1}{2}} = a_1 = 7$).
- For $a = [4, 8, 6, 0, 2]$, the median of a is 4 (since a after sorted is $[0, 2, 4, 6, 8]$, and $a_{\frac{n-1}{2}} = a_2 = 4$).
- For $a = [0, 100, 0]$, the median of a is 0.

Given an array b of n non-negative integers $b_0, b_1, \ldots, b_{n-1}$, where n is **odd**, and a non-negative integer x. Some elements of b are missing (indicated by $b_i = -1$). Recover the missing elements of b so that the median of the array b is exactly x, or report that it is impossible to do so.

Input

The first line contains a positive integer t ($1 \le t \le 10^4$) – the number of test cases. The description of each test case is as follows.

The first line contains two integers n and x ($1 \le n < 2 \cdot 10^5$, $0 \le x \le 10^9$) – the length of the array b and the required median value.

The second line contains n integers $b_0, b_1, \ldots, b_{n-1}$ ($-1 \le b_i \le 10^9$) – the elements of b (where $b_i = -1$ denotes that b_i is missing).

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

Output

For each test case, print `"YES"` (without quotes) if there is a way to recover the missing elements of b so that the median of b is x. Otherwise, print `"NO"` (without quotes).

If the answer is `"YES"`, on the second line, print n integers $b'_0, b'_1, \ldots, b'_{n-1}$ ($0 \le b'_i \le 10^9$), which represent the elements of array b after recovering the missing values. If there are multiple valid arrays b', print any of them. *It can be proven that under the constraints of this problem, if a valid array b' exists, there also exists an array b' where all elements are between 0 and 10^9.*

Sample Explanation

In the first test case, a possible recovery of b is $b' = [0, 4, 2]$, which has a median value of 2 (as the sorted b' is $[0, 2, 4]$).

In the second test case, a possible recovery of b is $b' = [4, 8, 6, 0, 2]$, which has a median value of 4 (as the sorted b' is $[0, 2, 4, 6, 8]$).

In the third test case, the only possible recovery of b is $b' = b$ itself, which has a median value of 8 (equal to x).

In the fourth test case, the only possible recovery of b is $b' = b$ itself, but it has a median value of 8 (which is different from $x = 9$).

In the fifth test case, it can be shown that the median of b' is always 4 regardless of the recovery, so no recovery of b exists where the median is 8.

Sample Input 1

```
5
3 2
0 -1 2
5 4
4 -1 6 0 -1
7 8
2 5 6 8 9 10 11
7 9
2 5 6 8 9 10 11
5 8
4 4 4 -1 -1
```

Sample Output 1

```
YES
0 4 2
YES
4 8 6 0 2
YES
2 5 6 8 9 10 11
NO
NO
```

Problem I: Inversing Palindrome

You are given two zero-indexed binary strings a and b, each string having length n. You can perform the following operation any number of times (possibly zero):

- Choose a palindromic substring of a with an even length, and flip all digits in the chosen substring (every digit `0` is turned into digit `1`, and vice-versa).

Determine whether it is possible to transform string a into string b.

Input

The first line contains a positive integer t $(1 \leq t \leq 10^4)$ – the number of test cases. The description of each test case is as follows.
The first line contains a positive integer n $(2 \leq n \leq 100)$ – the length of the two binary strings.
The second line contains a binary string a of length n.
The third line contains a binary string b of length n.
It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "YES" (without quotes) if you can transform string a into string b. Otherwise, print "NO" (without quotes).

Sample Explanation

In the first test case, a possible way to transform string a into string b is described as follows:

$$\texttt{\underline{1001}0} \xrightarrow{[0;3]} \texttt{011\underline{00}} \xrightarrow{[3;4]} \texttt{01111}$$

In the second test case, it is impossible to perform any operation on string a. Therefore, it is impossible to transform string a into a different string.
In the third test case, no operation is need to transform string a into string b.

Sample Input 1

```
3
5
10010
01111
3
101
110
4
0000
0000
```

Sample Output 1

```
YES
NO
YES
```

This page is intentionally left blank.

Problem J: Jemstones

Kosexy Biboo, one of the most renowned pebble collectors in the world, is celebrating her $700\,000$-th collected pebble! As a thank-you gift for everyone who has supported her career so far, Biboo plans to give away souvenirs to her patrons. But as pebbles are *too* precious and therefore not suited for gifting, she decided to use her other collection - *jemstones*.

There are $2 \cdot n$ patrons, each of whom will receive one gift from Biboo. Coincidentally, Biboo's *jemstone* collection consists of exactly $3 \cdot n$ green *jade jems* and $3 \cdot n$ red *jasper jems*. The $6 \cdot n$ *jemstones* are strung together to form a massive circular necklace and are numbered 1 to $6 \cdot n$ in clockwise order (i.e. the i-th *jemstone* is adjacent to the $(i + 1)$-th *jemstone*, and the n-th *jemstone* is adjacent to the first *jemstone*). Curiously, **no three consecutive *jemstones* are of the same kind**. Biboo wants to make $2 \cdot n$ gifts of the following two types (n gifts per type):

- **Type A** consists of a *jade jem*, a *jasper jem*, and a *jade jem* in that order.
- **Type B** consists of a *jasper jem*, a *jade jem*, and a *jasper jem* in that order.

Since removing *jemstones* from the collection and assembling them into a gift is time consuming, Biboo at any time can only choose three **consecutive** *jemstones* to make a gift. Note that after removing some *jemstones*, the remaining *jemstones* will become adjacent. Furthermore, since Biboo finds it difficult to keep track of the number of gifts of each type (she is learning how to count), she decided to make a gift of type A, then one of type B, then another of type A, and so on until all $6 \cdot n$ *jemstones* have been used.

Given the initial configuration of Biboo's *jemstone* collection, please help her find a way to make the required gifts, or tell her that it is impossible under her constraints.

Input

The first line contains a single integer t ($1 \leq t \leq 10\,000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n ($1 \leq n \leq 100\,000$).

The second line of each test case contains a string s of length $6 \cdot n$, describing Biboo's *jemstone* collection. The i-th character is `g` if the i-th *jemstone* is a green *jade jem*, and `r` if it is a red *jasper jem*.

For each test case, it is guaranteed that:

- There are exactly $3 \cdot n$ green *jade jems* and $3 \cdot n$ red *jasper jems*.
- No three consecutive *jemstones* are of the same kind.

It is guaranteed that the sum of n over all test cases does not exceed $100\,000$.

Output

For each test case, if it is not possible to make the required gifts under the constraints, output "NO" (without the quotes).

Otherwise, output "YES" (without the quotes). Then in the second line, output $2 \cdot n$ integers $a_1, a_2, \ldots, a_{2 \cdot n}$ ($1 \le a_i \le 6 \cdot n$), where a_i is the number of the first *jemstone* in clockwise order that Biboo needs to remove to make the i-th gift. Note that you should output the numbering of the *jemstone* in the **original** configuration, not its current position.

If there exists multiple ways to satisfy the constraints, output any of them.

Sample Explanation

In the first test case, Biboo's initial collection is as follows:

1	2	3	4	5	6
g	g	r	r	g	r

After removing the *jemstones* numbered 5, 6, and 1 to make a gift of type A, Biboo's collection is as follows:

2	3	4
g	r	r

Biboo can then remove the *jemstones* numbered 4, 2, and 3 to make a gift of type B. Note that choosing the *jemstones* numbered 2, 3, and 4 will result in a gift that consists of a *jade jem*, a *jasper jem*, and a *jasper jem*, which does not match type B.

In the second test case, Biboo cannot choose three consecutive *jemstones* to make a gift of type A from the initial collection.

Sample Input 1

```
2
1
ggrrgr
2
grrggrrggrrg
```

Sample Output 1

```
YES
5 4
NO
```

Problem K: Kettle Symphony

Henya is a genius. Everyone loves Henya, not only because she has an IQ of 199, but also because she can create the sound of boiling water kettles with her lovely voice, which is very pleasant to hear.

With her intelligence, Henya has many quirky but interesting ideas. On a boring day, Henya invented a music band with only kettles! Henya has n kettles, with some kettles currently turned on and some turned off. Henya said that when all the kettles are turned on, they will boil at the same time and then a perfect symphony will be played.

As a fan of Henya, you really want to hear this symphony. Therefore, Henya challenges you to turn on all the kettles with only the following operations (zero or more times):

- Choose **at least** a and **at most** b consecutive kettles and change the state of these kettles (turn on to turn off or vice versa).

Check if you can turn on all the kettles with the operations that Henya has proposed.

Input

The first line contains a positive integer t $(1 \le t \le 100)$ – the number of test cases. The description of each test case is as follows.

The first line contains three positive integers n, a, b $(1 \le a \le b \le n \le 100)$ – the number of Henya's water kettles and the minimum and maximum number of kettles that you can change the state in each operation.

The second line contains a binary string s of length n.

- $s_i = 0$ means that the i-th kettle is initially turned off,
- otherwise $s_i = 1$ means that the i-th kettle is initially turned on.

Output

For each test case, print "YES" (without quotes) if you can turn on all the kettles and enjoy the symphony. Otherwise, print "NO" (without quotes).

Sample Explanation

In the first test case, you can turn on the first 3 kettles, then turn on the last 4 kettles.

In the second test case, you can only choose all 3 kettles and change their states. When you change the state for the first time, you will get the state of the kettles as `010`. After the second time, you will get the state of the kettles as `101` – this is the initial state. Therefore, there is no way to turn on all the kettles.

In the third test case, you can perform the following operations, with the chosen kettles being underlined: $\texttt{\underline{10}1} \rightarrow \texttt{0\underline{11}} \rightarrow \texttt{\underline{000}} \rightarrow \texttt{111}$

Sample Input 1	Sample Output 1

```
3
7 3 4
0000000
3 3 3
101
3 2 3
101
```
	```
	YES
	NO
	YES
	```

Problem L: Lights Off

One evening, Bash and Chikapu arrived at an abandoned apartment with n floors. To their surprise, some of the lights on each floor still work! They ran up and down the apartment, turned on and off the lights and found that on the i-th floor, a_i lights are still working. On each floor, they number the lights from 1 to a_i, from left to right.

Currently, they are standing at the roof, after turning off all the lights on each floor except for the **right most** one (which they needed to go up the stairs).

Being competitive, they quickly came up with the following game. Two players take turns alternatively with Bash goes first. On each turn:

1. The player chooses a number i and go to the i-th floor.
2. The player then chooses a currently-turned-on light c on the i-th floor.
3. The player turns off the light c, and toggle the state of zero, one or two arbitrary lights, which are on the same floor and on the left of c

Bash always goes first, followed by Chikapu, and they both play optimally. The game ends when all lights are off, and who takes the latest turn is the winner. Your task is to determine which player will win if both play optimally. In case Bash (the first player) wins, count how many different first move Bash can take to guarantee his win, modulo $998\,244\,353$.

Two moves are considered different if:

- The chosen floor i is different, or
- The chosen light c is different, or
- The set of lights whose states are toggled is different

Input

The input consists of several test cases. The first line contains an integer t ($1 \le t \le 10^5$) – the number of test cases. The description of the test cases follows:

- The first line contains a single integer n ($1 \le n \le 10^5$), representing the number of floors.
- The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{18}$), where a_i is the number of lights on the i-th floor.

The sum of n over all the inputs do not exceed 10^5.

Output

For each test case, print the result in one line:

- Print either `Bash` or `Chikapu` - the winning player.
- If Bash wins, print an integer x - the number of first moves, after which Bash is guaranteed to win, modulo $998\,244\,353$.

Sample Input 1	Sample Output 1

2
3
1 2 1
4
1 2 3 4
	Bash 1
	Chikapu