National Collegiate Programming Contest (NCPC) 2023

Department of Computer Science and Engineering, Jahangirnagar University

Supported by

20 February, 2024

You get 14 Pages, 08 Problems & 240 Minutes

Sponsored By

Platinum Sponsors

BRAIN STATION 23 PLC.

NCC Bank
with you always

Gold Sponsors

DSi Dynamic Solution Innovators Ltd.

Other Sponsors

Therap (BD)

Problem A

Picturesque Skyline

The city you live in has a lot of high-rise buildings. Due to some city regulations, each building has a **unique height**. One day you went to the mountains outside the city. After a long hike you looked at the city to appreciate the beauty of it all. There you noticed something weird, actually two very weird things. The first being, from the right spot all the tall buildings look like they are on a single line. The other is that the buildings are forming multiple unusual groups of pyramid-like patterns with a small gap separating two adjacent groups.

A group of $2k + 1$ buildings, where k is a positive integer, forms a pyramid-like pattern. Assuming the heights of buildings are $h_1, h_2, \ldots, h_{2k+1}$ when sorted by position of the building from left to right, the group of building must satisfy $h_1 < h_2 < \ldots < h_k < h_{k+1} > h_{k+2} > \ldots > h_{2k} > h_{2k+1}$. So k buildings in increasing heights followed by the tallest building of the group which is followed by k buildings by decreasing heights. The city is formed of several such pyramid-like pattern groups. This makes the skyline of the city quite picturesque and unique.

Below are examples of two cities. The first one has a single pyramid-like pattern of 5 buildings. The second has 6 buildings in total with two pyramid-like patterns.

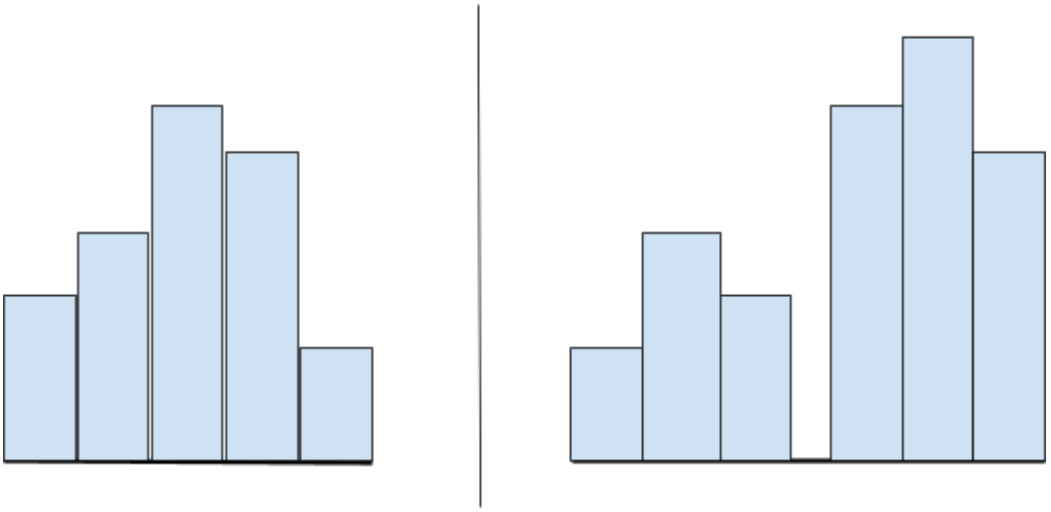

Fig: Two valid cities.

While looking at the skyline you got curious. How many different picturesque skylines can you get by arranging N buildings in one or more groups so that each group forms a pyramid-like pattern with a small gap between two adjacent groups? But there is a catch, if the number of buildings in any one group is too many then the skyline looks weird, so the maximum number of buildings in a group can't be more than K. Since the answer can be very big, answer it modulo 998244353. Please note that the actual heights of the buildings or the exact distance between two pyramid-like patterns doesn't matter, only the arrangements of the buildings and the existence of the gap between two adjacent pyramid-like pattern groups.

Input

The first line of the input contains an integer T $(1 \leq T \leq 1000)$ denoting the number of test cases. The next T lines each contains two integers N and K $(3 \leq K \leq N \leq 10^5)$ denoting the number of buildings and the maximum size of each group. It is guaranteed that $\Sigma N \leq 2 \cdot 10^5$ over all test cases and K is always odd.

Output

For each test case print one integer, the answer for the test case modulo 998244353.

Sample Input	Output for Sample Input
7 3 3 4 3 5 3 5 5 7 7 19 11 575 261	2 0 0 6 20 929394831 943612509

Explanation

For $N = 3\ and\ K = 3$, assuming the buildings have heights $h_1 < h_2 < h_3$, the two valid arrangements are $[h_1, h_3, h_2]$ and $[h_2, h_3, h_1]$.

For $N = 4$, there are no valid ways to arrange 4 buildings.

For $N = 5\ and\ K = 3$, similarly there aren't any valid ways.

For $N = 5\ and\ K = 5$, assuming the building have heights $h_1 < h_2 < h_3 < h_4 < h_5$, the valid arrangements are $[h_1, h_2, h_5, h_4, h_3]$, $[h_3, h_4, h_5, h_2, h_1]$, $[h_1, h_3, h_5, h_4, h_2]$, $[h_2, h_4, h_5, h_3, h_1]$, $[h_1, h_4, h_5, h_3, h_2]$, $[h_2, h_3, h_5, h_4, h_1]$.

Note that there is no way to have more than one group of pyramid-like patterns for this case.

Problem B

Easy String

There is an initially empty list, **B**. Throughout the process, you will be tasked with executing **Q** queries. Each query falls into one of three distinct types outlined below.

- **1 S f:** You have to append **f** copies of string **S** to the list **B**.
- **2 S f:** You have to remove **f** occurrences of string **S** from **B**. If the list **B** contains less than **f** copies of **S**, then remove all **S** from **B.**
- **3 S_1 S_2:** Determine the number of strings residing between strings S_1 and S_2 (inclusive) in the lexicographically sorted list of **B**.

Note that, each string provided in this problem will be given in *Run-Length Encoded* form of strings.

Run-length encoding (RLE) is a form of lossless compression. RLE is a simple method of compressing data by specifying the number of times a character repeats followed by the value of the character. For example, there is a string **S = "aaabbc"** in uncompressed form. If we compress the string it will be written as **S = a3b2c1.**

Each query of type 3 has an impact on the following queries of all types. Let's say the result of the last query of type 3 is **X.** Then, you get a query of type 1,2 or 3. You will replace each alphabet('a'-'z') with **(S[i] + X) mod 26.** Note that, you will not change the frequency of the characters. Remember that, initially **X** is 0.

All the strings we are describing here contain lowercase English alphabets only in their decompressed form.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 10)** denoting the number of test cases. After that, there will be **T** test cases formatted as below.

You will be given a positive integer **Q (1 ≤ T ≤ 5×10^5)** denoting the number of queries. Each of the next **Q** lines will contain queries in any of the three formats below.

- **1 S f**
- **2 S f**
- **3 S_1 S_2**

It is guaranteed that, **(1 ≤ f ≤ 10^5)** and there will be at least one query of type **3 S_1 S_2** in each test case.
Note that, the sum of length of all RLE compressed strings in the dataset will not exceed **10^6** over all test cases. Also note that, for each string a single character will not appear more than 10^5 times consecutively in their decompressed form.

Output

For each test case print “Case X:” first in a line, where **X** denotes the test case serial number. Then, print the answer for each query of type **3** in a new line.

Sample Input	Output for Sample Input
2 3 1 a1 2 2 a1 1 3 a1 a1 8 1 a3b1 4 3 a1 z1 1 w2x1 5 3 w1 v1 2 r3s1t1 4 3 r1 q1 2 r3s1 6 3 r1 q1	Case 1: 1 Case 2: 4 9 9 5

Problem C

Make A Beautiful Array

An array with at least **3** integers is called beautiful if its values increase at first, then decrease, and then increase and then decrease again, and so on. The beautiful array always has the **same** number of increasing and decreasing phases. So, there are always some peak values in the array where the immediate left value and right value of each peak value are less than the peak value. For example, [1, 3, 1, 2, 4, 2, 1] is a beautiful array with 3 and 4 as the peak values. Each beautiful array has its own beauty factor. The beauty factor of a beautiful array is the number of elements divided by the number of peak values in the array.

You are given an array **A** of **n** integers. Suppose you choose some elements from the array, and the chosen elements maintaining the order given in array **A** create another array **B**. You have to create the array **B** such that **B** is a beautiful array with the maximum beauty factor possible.

Input

The first line will contain a single integer **T ($1 \le T \le 10^4$)**. Each test case will have a single integer **n ($3 \le n \le 5 \cdot 10^5$)**, denoting the number of elements in array **A**. The second line of each test case will contain **n** space-separated integers.

Output

For each test case, print "**Case x: y**" where **x** is the test case number and **y** is the beauty factor of the array **B** if you can make it. If you can't make such an array, then **y** is **-1**.

Constraints

- **$1 \le T \le 10^4$**
- **$3 \le n \le 5 \cdot 10^5$**
- **$-10^{12} \le A[i] \le 10^{12}$**
- **the sum of n over all test cases doesn't exceed 5×10^5**

Sample Input	Output for Sample Input
2 5 5 2 1 3 4 7 1 2 3 2 1 2 3	Case 1: -1 Case 2: 5

Problem D

Battle of Boberland

The greatest kingdom of Boberland was ruled by King Bob, a tactical genius renowned for his strategic brilliance. He commands a vast army, with each soldier possessing unique fighting skills and strength. Before each battle, King Bob meticulously researched his enemy and selected specific subsegments of his army with enough skill and strength to ensure victory. His success led to triumphs in over a hundred battles, solidifying his reputation as an extraordinary warrior.

King Lucian, driven by insatiable greed, sought to conquer Boberland and seize its wealth. After vanquishing a neighboring kingdom, he set his sights on King Bob's domain. Lucian's spies informed him of Bob's strategy, revealing that Lucian's army could only win if the bitwise ***XOR*** of their total strength with each soldier in Bob's selected subsegments resulted in a value less than or equal to Lucian's total army strength. However, Lucian's forces lacked knowledge of which soldiers Bob would deploy, as the king customarily chose his warriors at the last moment based on enemy analysis.

Time is of the essence, and King Lucian cannot afford the delay of calculating all possible army configurations Bob might use. As one of Lucian's most cunning soldiers, you are tasked with infiltrating the enemy ranks to determine if victory is attainable. Due to the looming battle, Lucian needs you to calculate the minimum strength required for specific subsegments of Bob's army to ensure the win. Here's your mission in detail:

You are given an array of length n representing the strengths of Bob's army. Each element a_i denotes the strength of the i^{th} soldier. King Lucian wants to know the minimum strength required to defeat Bob's Q most important subsegment of the army. For each Q subsegment, you have to find a minimum non-negative value x which satisfies the below condition:

$\forall i$, $a[i] \text{ XOR } x \leq x$
$l \leq i \leq r, 1 \leq l \leq r \leq n$
l, r belongs to the leftmost and rightmost index of the given subsegment.

N.B: A bitwise XOR is a binary operation that takes two bit patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if only one of the bits is 1, but will be 0 if both are 0 or both are 1. The bitwise XOR of 55 and 34 is 23.

Input

The first line will contain a single integer T $(1 \leq T \leq 50)$. Each test case will contain two positive integers N $(1 \leq N \leq 3 \cdot 10^5)$ and $Q(1 \leq Q \leq 3 \cdot 10^5)$, denoting the size of the array and number of queries respectively. The next line will contain N non-negative integers a_i $(0 \leq a_i \leq 10^9)$ Each of the following Q lines will contain two integers L and R, denoting the subsegment of the array. For each query, you have to calculate the minimum non-negative integers which satisfy the described condition above.

Total number of N over all test cases will not exceed $3 \cdot 10^5$ and total number Q over all test cases will not exceed $3 \cdot 10^5$.

Output

Print the case number in a single line followed by the calculated strength for every query in desired format. Please see the sample for details.

Sample Input	Output for Sample Input
2 6 5 2 31 21 29 8 11 1 6 1 4 2 5 3 6 5 6 10 7 59734171 371 166823139 2559 740 656957380 852151104 2376 972816575 2719 1 7 3 10 2 3 5 7 6 9 4 6 1 8	Case 1: 26 18 24 24 8 Case 2: 704645888 671091200 134217984 536871424 536872960 536873472 704645888

Problem E

Platonic number in a tree

You are given a rooted tree consisting of **n** nodes numbered from **1** to **n**, each node **i** having value a_i **(1 ≤ i ≤ n)**. The root of the tree is node **1.**

For each node **u** from **1** to **n**, calculate the Platonic number of node u as following:

- Consider the path from node **u** to node **1**. Let it be $v_1, v_2, v_3, \ldots, v_k$. Where $v_1 = u$ and $v_k = 1$ and node v_{i+1} is the parent of node v_i for **1 ≤ i < k**.
- Consider the array **b** of size **k** as $a_{v_1}, a_{v_2}, a_{v_3}, \cdots, a_{v_k}$ so $b_i = a_{v_i}$ for **1 ≤ i ≤ k**.
- Consider the array **c** of size **k** as prefix-max array of **b**. So $c_1 = b_1$ and $c_i = \max(b_i, c_{i-1})$ for **1 < i ≤ k**
- Platonic number of node **u** is the sum of the elements of the array **c**.

Input

The first line will contain a single integer **t (1 ≤ t ≤ 1e5)**, denoting the number of test cases. Then **t** test cases follow.

First line of each test case will have a single integer **n (1 ≤ n ≤ 5e5)**, denoting the number of nodes.

Second line of each test case will contain **n** integers $a_1, a_2, a_3, \ldots a_n$ **(1 ≤ a_i ≤ 1e9)**, denoting the values associated with each node.

The following **n-1** lines will contain two integers **u** and **v (1 ≤ u, v ≤ n, u ≠ v)** which describe a pair of nodes connected by an edge. It is guaranteed that the given graph is a tree and has no loop or multiple edges.

It is guaranteed that the sum of **n** for all test cases does not exceed **5e5.**

Output

For each test case, print **n** number: the platonic number of a node **u**, for each node from **1** to **n**. Print the case number in a single line followed by the line containing **n** numbers separated by space. Please see the sample for details.

Sample Input	Output for Sample Input
2 7 7 6 6 1 5 2 10 1 2 1 3 2 4 2 5 3 6 3 7 3 3 1 2 1 2 2 3	Case 1: 7 13 13 14 18 15 30 Case 2: 3 4 7

Problem F

Aragorn is back

Aragorn, the mighty king, has secretly trained a hidden army of numbered soldiers (**1** to **N**). Each soldier only knows the strength of their neighbors(i.e. **i^{th}** soldier knows **$(i-1)^{th}$** and **$(i+1)^{th}$** soldiers only). After training, Aragorn wants a powerful, **connected squad** that may contain one or multiple soldiers. They're connected if they know each other directly or indirectly (i.e. 1 knows 2, 2 knows 3, so 1 knows 3).

Aragorn wants the strongest squad that follows the following criteria:

1. The squad has the **maximum overall strength sum**.
2. As resources are minimal to prepare new weapons for the squad, he prefers the **fewest number of soldiers**, ensuring maximum strength.

He has prepared **customized weapons** for the squad based on the strength of each soldier and then sends the squad to the battlefield.

He can not prepare any more weapons but keeps building new squads **from all the N soldiers** in the same way and sends them to any upcoming battles with the weapons prepared earlier. He sends one squad to the battlefield at a time and each squad must follow all the criteria.

Aragorn wonders about the maximum number of soldiers that can attend at least one battle if he optimally prepares the weapons for the initial squad. Can you help him figure it out?

N.B. If a customized weapon is prepared for a soldier having strength x, then this weapon can only be used by any soldier having strength x.

Input

The first line will contain a single integer ***T (1 ≤ T ≤ 40)***. Each test case will contain one positive integer ***N (1 ≤ N ≤ $3 \cdot 10^5$)*** denoting the size of the array. The next line will contain ***N*** integers ***A_i ($-10^6 \le A_i \le 10^6$)*** representing the strength of the ***i^{th}*** soldier. The summation of ***N*** over all the test cases will not exceed ***$3 \cdot 10^6$.***

Output

Print the case number followed by the maximum number of soldiers that can attend at least one battle in a single line. Please see the sample for details.

Sample Input	Output for Sample Input
2 5 1 2 3 4 -5 18 1 4 -5 1 4 -10 2 -2 4 1 -12 1 2 2 -10 1 2 2	Case 1: 4 Case 2: 6

Problem G

Game of Removing

You will be given two arrays **A** and **B**, each containing **N** number of integers. Now you have to make B as a subsequence of **A** by removing at most one subarray or possibly none from **B**. But you have to make sure that the size of the subarray is as minimum as possible.

For more clarification see the Sample **I/O** and Explanation.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.
Each test case starts with an integer **N (≤ 10^3)** . Then the next line contains **N** integers, representing array **A**. Then also the next line contains **N** integers that represent array **B**.

$0 \le A_i \le 10^9$
$0 \le B_i \le 10^9$
$1 \le i \le N$

Output

For each case, print the case number and the minimum possible length of the deleted subarray.

Sample Input	Output for Sample Input
4 5 2 7 0 2 1 2 3 0 4 7 3 3 5 7 1 7 9 4 10 19 27 35 10 19 27 35 2 1 2 3 4	Case 1: 3 Case 2: 3 Case 3: 0 Case 4: 2

Explanation:
1st Case: After Deleting [3, 0, 4], B becomes [2, 7] which is a subsequence of A = [2, 7, 0, 2, 1].
2nd Case: After Deleting [1, 7, 9], B becomes [] which is a subsequence of A = [3, 5, 7]. You may safely assume that an empty array is a subsequence of any other array.
3rd Case: there is no need to delete any element from B = [10, 19, 27, 35], which is already a subsequence of A = [10, 19, 27, 35].
4th Case: After Deleting [3, 4], B = [] which is a subsequence of A = [1, 2].

Problem H

Calendar (Please No More)

Calendar for Month of February, 2024

Sun	Mon	Tue	Wed	Thu	Fri	Sat
-	-	-	-	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	-	-

You are given a calendar with a table of 6 rows and 7 columns. The columns are labeled as Sun, Mon, Tue, Wed, Thu, Fri, and Sat. Given a date in the format DD/MM/YYYY and the corresponding day name, your task is to print the calendar for the given month.

Assume January is the 1st month, and December is the 12th month. The number of days in each month is as follows: January (31 days), February (28 days unless it's a leap year, in which case it has 29 days), March (31 days), April (30 days), May (31 days), June (30 days), July (31 days), August (31 days), September (30 days), October (31 days), November (30 days), and December (31 days). A leap year occurs every four years, and any year that is evenly divisible by 100 is not a leap year unless it is also evenly divisible by 400.

In this problem you have to print the calendar of the given month.

Input

The first line contains an integer T ($1 \leq T \leq 100$), the number of test cases.
For each test case, a line contains a string "DD/MM/YYYY DayName" representing the date and day name. See the sample I/O for more clarity.

$01 \leq DD \leq 31$
$01 \leq MM \leq 12$
$1000 \leq YYYY \leq 9999$
DayNames are (Sun, Mon, Tue, Wed, Thu, Fri, Sat).

Output

For each test case, print the calendar for the given month in a tabular format with 5 rows and 7 columns, where each column represents a day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat). Each cell in the table should display the corresponding day of the month.

The output should be structured as follows:

- Days of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat) should be labeled above the columns.
- Each row should represent a week, and each cell should contain the corresponding day of the month.
- Blank cells before the 1st day and after the last day of the given month should be filled with dashes (" - ").
- If it is not possible to print all the days of a month within the five weeks, then the remaining dates get printed from the first week of that month.
- Print a blank line after each test case.

See the sample I/O section for more clarity.

Sample Input

```
2
20/02/2024 Tue
01/01/2000 Sat
```

Output for Sample Input

```
|---------------------------|
|Sun|Mon|Tue|Wed|Thu|Fri|Sat|
|---------------------------|
| - | - | - | - |  1|  2|  3|
|---------------------------|
|  4|  5|  6|  7|  8|  9| 10|
|---------------------------|
| 11| 12| 13| 14| 15| 16| 17|
|---------------------------|
| 18| 19| 20| 21| 22| 23| 24|
|---------------------------|
| 25| 26| 27| 28| 29| - | - |
|---------------------------|

|---------------------------|
|Sun|Mon|Tue|Wed|Thu|Fri|Sat|
|---------------------------|
| 30| 31| - | - | - | - |  1|
|---------------------------|
|  2|  3|  4|  5|  6|  7|  8|
|---------------------------|
|  9| 10| 11| 12| 13| 14| 15|
|---------------------------|
| 16| 17| 18| 19| 20| 21| 22|
|---------------------------|
| 23| 24| 25| 26| 27| 28| 29|
|---------------------------|
```