Qualification Round Contest TSE-2026 S2

Contest Information

- **Duration:** 2.30 Hours
- **Contest Style:** ICPC Format

Important Notice

- The decision of the **Divine IT Judges** shall be **final and binding** in all matters related to the contest.
- **Use of Artificial Intelligence tools is strictly prohibited.**
- Any submission found to be **suspicious or violating contest rules** will result in **immediate disqualification**.
- Such violations may also lead to **banning the candidate from future job opportunities at Divine IT Limited**.

	Problem A	
	Caravan Across the Kingdom	

In the ancient Kingdom of Aleria, a royal caravan master named Sir Rowan is responsible for transporting groups of travelers between cities. The kingdom consists of several cities connected by two-way trade roads.

Each road is served by a caravan wagon that travels only between the two cities it connects. Due to safety and weight restrictions, every wagon has a maximum passenger capacity.

Sir Rowan has been assigned a mission:
He must transport a group of travelers from one city to another. However, the caravan wagons cannot always carry all travelers in a single journey. Therefore, Sir Rowan may need to make multiple trips, possibly using the same route several times.

Sir Rowan owns a detailed map showing:

- The cities and the roads connecting them
- The maximum number of passengers each wagon can carry

His goal is to choose the route that allows the largest number of travelers per trip, so that the total number of trips is minimized.

Unfortunately, planning the optimal route is not his strength so he asks you for help.

Your Task

For each mission:

- Determine the maximum number of travelers that can be transported in one trip along any route from the source city to the destination city.
- Based on this, compute the minimum number of trips required to transport all travelers.

Input Format

The input consists of multiple test cases.

- The first line of each test case contains two integers:
  - N ($1 \le N \le 100$): the number of cities
  - R: the number of roads
- The next R lines each contain three integers:

- C1, C2: cities connected by a road
- P (P > 1): maximum passenger capacity of the wagon on that road

- The next line contains three integers:
  - S: starting city
  - D: destination city
  - T: total number of travelers
- The input ends with a line containing two zeroes (`0 0`).

Output Format

For each test case:

1. Print the scenario number.
2. Print the minimum number of trips required.
3. Print a blank line after each test case.

Sample Input	Sample Output
7 10 1 2 30 1 3 15 1 4 10 2 4 25 2 5 60 3 4 40 3 6 20 4 7 35 5 7 20 6 7 30 1 7 99 0 0	Scenario #1 Minimum Number of Trips = 5

Explanation

- Now, if he wants to take 99 tourists from city 1 to city 7, he will require at least 5 trips and the route he should take is : 1 - 2 - 4 - 7.

	Problem B	
	Shared Study Desks	

A public library has N study desks available for visitors.
At the beginning of the day, all desks are unused.

During the day, visitors arrive and request a desk to study.
If a visitor already has a desk, they may continue using the same desk for their next request.
If a visitor does not currently have a desk, they must be assigned one.

Assigning a desk has a cost:

- When a desk is assigned to a new visitor, it must be prepared.
- If all desks are occupied, the librarian may ask any visitor to leave, making a desk available for someone else.

Fortunately, the librarian knows the exact order of visitor requests in advance.

Your task is to decide how to assign desks so that the total number of desk preparations is minimized.

Task

For each test case, determine the minimum number of times a desk needs to be prepared in order to serve all visitor requests.

Input Format

- The first line contains an integer T, the number of test cases.
- For each test case:
  - The first line contains two integers N and M:
    - N: number of desks
    - M: number of visitor requests
  - The next line contains M integers, representing the sequence of visitor requests.
    - Each integer represents a visitor ID.

Output Format

- For each test case, output a single integer:
  the minimum number of desk preparations required.

Constraints

- 1 ≤ T ≤ 100
- 1 ≤ N ≤ 200
- 1 ≤ M ≤ 400
- 1 ≤ Visitor ID ≤ 400

Sample Input

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

Sample Output

```
4
2
4
4
```

Explanation (Example Case 1)

Initially, both study desks are empty.

- Visitor 1 arrives and is assigned a desk. One desk is prepared.
- Visitor 2 arrives and takes the remaining free desk. A second preparation is needed.
- Visitor 3 arrives, but all desks are occupied. The librarian asks either visitor 1 or 2 to leave, prepares the freed desk, and assigns it to visitor 3.
- Visitor 4 arrives next. Again, all desks are occupied, so one current visitor must leave. The freed desk is prepared and assigned to visitor 4.

In total, the desks are prepared 4 times, which is the minimum possible.

	Problem C		
	The Waiting Line Monitor		

At a large service center, people wait in a single line to be served.
Each person in the line has a unique ID number.

The system works as follows:

- Initially, there are N people already standing in the line.
- During the day, two types of events can happen:
    1. A new person joins at the end of the line
    2. The staff calls the k-th person from the front of the line to be served

When a person is called, they leave the line permanently.

You have access to the entire sequence of events in advance, and your task is simply to report who gets called each time.

Task

For each test case:

- Maintain the line of people.
- For every “call” operation, output the ID of the person being called.
- If the requested position does not exist, output `none`.

Input Format

- The first line contains an integer T (T ≤ 5), the number of test cases.

For each test case:

- A blank line
- One line with two integers:
    - n number of people initially in the line ($0 \le n \le 10^5$)
    - q number of operations ($1 \le q \le 50{,}000$)
- One line with n space-separated integers, representing the IDs of people in the line (from front to back)
- The next q lines contain one of the following operations:

- a p → add a person with ID p to the end of the line
- c k → call the k-th person from the front

All IDs are positive 32-bit signed integers.

Output Format

For each test case:

- Print the case number.
- For every `c k` operation:
  - Print the ID of the called person
  - Or print `none` if fewer than `k` people are in the line

Sample Input	Sample Output
2 5 5 6 5 3 2 1 c 1 c 1 a 20 c 4 c 4 2 1 18811 1991 c 1	Case 1: 6 5 20 none Case 2: 18811

Explanation (Case 1)

Initial line:

`[6, 5, 3, 2, 1]`

- Call 1st person → 6 leaves
- Call 1st person → 5 leaves
- Add person 20 → line becomes `[3, 2, 1, 20]`
- Call 4th person → 20 leaves
- Call 4th person → no such person → none

Problem D

Palindrome by Deletion

A data analyst is examining words and becomes fascinated by palindromes strings that read the same forward and backward.

Given a word, the analyst wonders:

> *In how many different ways can letters be removed so that the remaining letters form a palindrome?*

Some important observations:

- Letters may be removed from any positions.
- The order of removals does not matter only the final remaining string matters.
- It is allowed to remove no letters at all, if the original word is already a palindrome.

Your task is to count the total number of distinct ways to remove letters so that the resulting string is a palindrome.

Task

For each test case:

- Given a word W, determine the number of distinct subsequences of W that form a palindrome.

Input Format

- The first line contains an integer T (T ≤ 200), the number of test cases.
- Each test case consists of a single word W.
    - 1 ≤ length(W) ≤ 60
    - The word contains only uppercase English letters.

Output Format

For each test case, print:

- `Case X: Y`

Where:

- X is the test case number (starting from 1)
- Y is the number of ways to remove letters so that the remaining string is a palindrome

Sample Input	**Sample Output**

`3`
`SALADS`
`PASTA`
`YUMMY`
	`Case 1: 15`
	`Case 2: 8`
	`Case 3: 11`

Explanation (Case 2: "PASTA")

There are 8 distinct ways to remove letters so that the remaining string is a palindrome:

1. Remove `PAST_` → remaining string is `A`
2. Remove `PAS_A` → remaining string is `T`
3. Remove `PA_TA` → remaining string is `S`
4. Remove `P_STA` → remaining string is `A`
5. Remove `_ASTA` → remaining string is `P`
6. Remove `P_ST_` → remaining string is `AA`
7. Remove `P__T_` → remaining string is `ASA`
8. Remove `P_S__` → remaining string is `ATA`

Each remaining string is a palindrome, and each distinct result is counted exactly once.

Problem E

Clear the Sequence

You are given a string consisting of only the letters A and B.
You can perform the following operation multiple times:

- Choose a subsequence of the string in which no two consecutive letters are the same (i.e., it alternates between A and B).
- Delete all letters of that subsequence from the string.

Your goal is to remove all letters from the string in the minimum number of operations.

Task

Given a string S, determine the minimum number of operations needed to remove all its characters using the rule above.

Input Format

- The first line contains an integer T the number of test cases.

For each test case:

- First line: integer N the length of the string
- Second line: string S of length N, containing only characters A and B

Output Format

For each test case, output a single integer on a new line:

- The minimum number of operations required to delete all characters from the string.

Constraints

- $1 \le T \le 10^4$
- $1 \le N \le 2 \times 10^5$
- Sum of all N across test cases $\le 2 \times 10^5$
- $S_i \in \{A, B\}$

Sample Input	Sample Output
2 4 ABAB 3 BBB	1 3

Explanation

Test Case 1:

- The string is `ABAB`.
- The whole string is already alternating, so we can remove it in 1 operation.

Test Case 2:

- The string is `BBB`.
- All letters are the same, so we can only remove one letter at a time.
- Minimum operations = 3.

	Problem F	
	Magical Digit Sum	

In a magical kingdom, numbers have special powers. For any non-negative integer `N`, a wizard defines:

- S(N) = sum of the odd digits of `N` + twice the sum of the even digits of `N`.
- D(N) = last digit of S(N) (i.e., S(N) mod 10).

For example:

- S(5) = 5 → D(5) = 5
- S(456) = 2×4 + 5 + 2×6 = 25 → D(456) = 5
- S(314159) = 3 + 1 + 2×4 + 1 + 5 + 9 = 27 → D(314159) = 7

The wizard wants to know, given two numbers A and B, the sum of D(N) for all integers N from A to B, inclusive.

Task

For each test case:

- Compute the sum of D(N) for all integers between A and B, including both endpoints.

Input Format

- The first line contains an integer T the number of test cases.
- Each of the next T lines contains two integers A and B (0 ≤ A ≤ B ≤ 400,000,000).

Output Format

- For each test case, output a single integer: the sum of D(N) from A to B.

Constraints

- $1 \le T \le 1000$
- $0 \le A \le B \le 400{,}000{,}000$

Sample Input	Sample Output
3 1 8 28 138 314159 314159	36 495 7