**Minimum Deletions: -**

**Medium Accuracy: 58.8% Submissions: 39K+ Points: 4**

Given a string of **S** as input. Your task is to write a program to **delete**the minimum number of characters from the string so that the resultant string is a palindrome.  
**Note:** The order of characters in the string should be maintained.

**Example 1:**

**Input:**S= **"**aebcbda"

**Output:**   
2

**Explanation**:   
Remove characters 'e' and 'd'.

**Example 2:**

**Input**:   
S = "geeksforgeeks"

**Output:**   
8

**Explanation**:   
One of the possible result string can be "eefee", so answer is 13 - 5 = 8.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **minimumNumberOfDeletions()** which takes the string **S**as inputs and returns the minimum number of deletions required to convert **S** into a pallindrome.  
  
**Expected Time Complexity:** O(|S|2)  
**Expected Auxiliary Space:** O(|S|2)  
  
**Constraints:**  
1 ≤ |S| ≤ 103

**Code: -**

//{ Driver Code Starts

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

//User function template for C++

class Solution{

public:

int n;

int lcss(string &ori, string &rev, int i, int j, vector<vector<int>> &dp){

// base case

if(i >= n or j >= n)

return 0;

// dp found case

if(dp[i][j] != -1)

return dp[i][j];

// recursive case

int ans = 0;

if(ori[i] == rev[j])

ans = 1 + lcss(ori, rev, i+1, j+1, dp);

else

ans = max(lcss(ori, rev, i, j+1, dp),

lcss(ori, rev, i+1, j, dp));

// return from curretn state

return dp[i][j] = ans;

}

int minimumNumberOfDeletions(string S) {

// code here

n = S.size();

vector<vector<int>> dp(n, vector<int>(n, -1));

string original = S;

reverse(S.begin(), S.end());

return n - lcss(original, S, 0, 0, dp);

}

};

//{ Driver Code Starts.

int main(){

int t;

cin >> t;

while(t--){

string S;

cin >> S;

Solution obj;

cout << obj.minimumNumberOfDeletions(S) << endl;

}

return 0;

}

// } Driver Code Ends

**T.C: - O(|S|2)**

**S.C: - O(|S|2)**