**Longest Palindrome in a String :-**

Medium Accuracy: 23.2% Submissions: 250K+ Points: 4

Given a string S, find the longest palindromic substring in S.**Substring of string S:** S[ i . . . . j ] where 0 ≤ i ≤ j < len(S)**. Palindrome string:** A string that reads the same backward. More formally, S is a palindrome if reverse(S) = S.**In case of conflict**, return the substring which occurs first ( with the least starting index).

**Example 1:**

**Input:**

S = "aaaabbaa"

**Output:** aabbaa

**Explanation**: The longest Palindromic

substring is "aabbaa".

**Example 2:**

**Input**:

S = "abc"

**Output:** a

**Explanation**: "a", "b" and "c" are the

longest palindromes with same length.

The result is the one with the least

starting index.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **longestPalin()**which takes the string S as input and returns the longest palindromic substring of S.

**Expected Time Complexity:**O(|S|2).  
**Expected Auxiliary Space:**O(1).

**Constraints:**  
1 ≤ |S| ≤ 103

**Code :-**

//{ Driver Code Starts

#include<bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution {

public:

string longestPalin (string &s) {

int n = s.size();

if(n==1) return s;

vector<char> help(n+n+1);

vector<int> length(n+n+1);

help[0] = '|';

int ind=1;

for(auto i:s){

help[ind]=i; ++ind;

help[ind]='|'; ++ind;

}

int center=0, radius=0;

while(center < n+n+1){

while(center-(radius+1)>=0 && center+(radius+1)<n+n+1 &&

help[center-(radius+1)]==help[center+(radius+1)])

radius += 1;

length[center] = radius;

int oldcenter = center, oldradius=radius;

center += 1;

radius = 0;

while(center <= (oldcenter+oldradius)){

int mirroredcenter = oldcenter - (center - oldcenter);

int maxmirroredradius = oldcenter + (oldradius - center);

if(length[mirroredcenter] < maxmirroredradius){

length[center] = length[mirroredcenter];

center += 1;

}

else if(length[mirroredcenter] > maxmirroredradius){

length[center] = maxmirroredradius;

center += 1;

}

else{

radius = maxmirroredradius;

break;

}

}

}

int ans=INT\_MIN, ansind;

for(int i=0; i<n+n+1; i++){

if(length[i]>ans){

ans = length[i];

ansind = i;

}

}

string final="";

if(help[ansind]!='|')

final = final + help[ansind];

ind=1;

while(ans--){

if(help[ansind-ind]!='|')

final = help[ansind-ind] + final + help[ansind+ind];

++ind;

}

return final;

}

};

//{ Driver Code Starts.

int main()

{

int t; cin >> t;

while (t--)

{

string S; cin >> S;

Solution ob;

cout << ob.longestPalin (S) << endl;

}

}

// Contributed By: Pranay Bansal

// } Driver Code Ends

**T.C :- O(n)**

**S.C :- O(n)**