```cpp
/*
        Given a string S. Find how many distinct substring of S which is palindrome.
        Number of Test case, T<=20. Each test case consists of a string S,
        whose length is less than 100000 and only contains lowercase letters.
*/
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 100005;
struct Node{
    int nxt[26];
    int length, suffixLink;
    int startPos, endPos;
};

struct PalTree{
    Node tree[MAXN];
    Node root1, root2;
    int ptr, curNode;
    char s[MAXN];

    void init(){
        root1.length = -1, root1.suffixLink = 1;
        root2.length =  0, root2.suffixLink = 1;
        tree[1] = root1, tree[2] = root2;
        ptr = curNode = 2;
    }

    int addLetter(int pos){
        int ch = s[pos]-'a';
        int cur = curNode;

        while(true){
            int curLength = tree[cur].length;
            if(pos-1-curLength >= 0 && s[pos-1-curLength] == s[pos])break;
            cur = tree[cur].suffixLink;
        }

        if(tree[cur].nxt[ch] != 0){
            curNode = tree[cur].nxt[ch];
            return 0;
        }

        ptr++;
        curNode = ptr;
        tree[cur].nxt[ch] = curNode;
        tree[curNode].length = tree[cur].length + 2;
        tree[curNode].startPos = pos - tree[curNode].length + 1;
        tree[curNode].endPos = pos;

```

```
        if(tree[curNode].length == 1){
            tree[curNode].suffixLink = 2;
            return 1;
        }

        while(true){
            cur = tree[cur].suffixLink;
            int curLength = tree[cur].length;
            if(pos-1-curLength >= 0 && s[pos-1-curLength] == s[pos]){
                tree[curNode].suffixLink = tree[cur].nxt[ch];
                break;
            }
        }

        tree[curNode].suffixLink = tree[cur].nxt[ch];
        return 1;
    }

    void Clear(){
        for(int i=0; i<=ptr; i++){
            memset(tree[i].nxt, 0, sizeof(tree[i].nxt));
        }
    }
};

PalTree Pt;

int main()
{
    int tt; scanf("%d",&tt);
    for(int ks=1; ks<=tt; ks++){
        scanf("%s",&Pt.s);
        int n = strlen(Pt.s);
        Pt.init();
        int ans = 0;
        for(int i=0; i<n; i++){
            ans += Pt.addLetter(i);
        }
        printf("Case #%d: %d\n",ks,ans);
        Pt.Clear();
    }
    return 0;
}

Input           Output
3
aaaa            Case #1: 4
abab            Case #2: 4
abcd            Case #3: 4
```