```
/* Given a string S of lower-case Latin letters(a-z), |S| <= 100000.
Find the the longest contiguous palindromic substring of S.
If there are several such strings you should output the first of them. */
#include<bits/stdc++.h>
using namespace std;
#define ll long long
const int MAXN = 100005;
struct Node{
    int nxt[52];
    int val;
    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;
    }

    void addLetter(int pos){
        int ch;
        if(s[pos]>='A'&&s[pos]<='Z') ch = s[pos]-'A';
        else ch = s[pos]-'a'+26;
        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];
            tree[curNode].val++;
            return;
        }

        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;
            tree[curNode].val = 1;
            return;
        }

        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];
        tree[curNode].val = 1;
        return;
    }

    void getResult(){
        int maxx = 0;
        int Start = 0, End = 0;
        for(int i=3; i<=ptr; i++){
            if(tree[i].length>maxx){
                maxx = tree[i].length;
                Start = tree[i].startPos;
                End = tree[i].endPos;
            }
        }

        for(int i=Start; i<=End; i++) printf("%c",s[i]);

        printf("\n");
    }

    void Clear(){
        for(int i=0; i<=ptr; i++){
            memset(tree[i].nxt, 0, sizeof(tree[i].nxt));
        }
    }
};
PalTree Pt;
int main(){
    scanf("%s",&Pt.s);
    int n = strlen(Pt.s);
    Pt.init();
    for(int i=0; i<n; i++) Pt.addLetter(i);
    Pt.getResult();
    return 0;
}
```