```
#include<stdio.h>
#include<iostream>
#include<string.h>
#include<algorithm>
#define M 1000010
using namespace std;
char str[M],str_new[2*M];     //原串和辅助串
int p[2*M],len;                //p[i]表示以第 i 个字符为中心的最长回文的半径;字符串长度 len
void init()                    //构造辅助串
{
    len=strlen(str);           //计算字符串长度
    str_new[0]='@';            //辅助串的首字符
    str_new[1]='#';            //辅助串的间隔字符
    for(int i=0; i<len; i++)   //逐个字符地构造辅助串
    {
        str_new[i*2+2]=str[i];
        str_new[i*2+3]='#';
    }
    str_new[len*2+2]='$';      //辅助串的尾字符
}
void Manacher()                //计算和输出最长回文的半径
{
    memset(p,0,sizeof(p));     //p[i]表示以第 i 个字符为中心的最长回文的半径，所有最长回文的半径初始化为 0
    int mx=0,di,ans=0;         //以 id 为中心的最长回文的右边界为 mx，即 mx=id+p[id]，mx 和最长回文的长度 ans 初始化为 0
    for(int i=1; i<len*2+2; i++)  //枚举每一个可能的中心字符
    {
        if(mx>i)               //根据 i 位置在 mx 位置的左侧还是右侧，调整以最长回文的半径的最长回文半径的初始值
            p[i]=min(mx-i,p[di*2-i]);
        else
            p[i]=1;
        for(; str_new[i-p[i]]==str_new[i+p[i]]; p[i]++); //以 i 位置为中心计算最长回文半径 p[i]
        if(p[i]+i>mx)          //若以 i 为中心的右边界大于 mx，则中心 id 调整为 i，重新计算右边界 mx
            mx=p[i]+i,di=i;
```

```
            ans=max(ans,p[i]);          //调整最长回文的半径
        }
        printf("%d\n",--ans);           //输出最长回文的半径
}
int main()
{
        int t=1;                        //测试用例编号初始化
        while(~scanf("%s",str))         //反复输入字符串，直至输入"END"为止
        {
            if(!strcmp(str,"END")) break;
            printf("Case %d: ",t++);    //输出测试用例编号
            init();                     //构造辅助串
            Manacher();                 //计算和输出最长回文的半径
        }
}
```