```
#include <cstdio>
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
typedef long long ll;
char s[1001000];                    // 输入字串
int mod=10009;                      //模
int len,k=131;                      //s 的长度为 len
ll hash[1001000];                   // hash[i]存储以第 i 个字符为尾的前缀的散列值
ll cal(int x,ll y)                  //计算和返回 y^x % mod 的结果值
{
    ll re=1;                        //结果值初始化
    while(x)                        //分析次幂 x 的每一个二进制位
    {
        if(x&1) re=(re*y)%mod;      //若当前位为 1，则累乘当前位的权并取模
        x>>=1;y=(y*y)%mod;          //次幂 x 右移一位，计算该位的权后取模
    }
    return re;                      //返回结果值
}
bool check(int x)                   //若所有长度为 x 的相邻子串对应的散列函数值相等，
则返回 true；否则返回 false
{
    ll cc=cal(x,(ll)k);             //计算 k^x % mod
    for(int i=(x<<1);i<=len;i+=x)   //搜索字符 i（2*x≤i≤len）。若任一长度 i 的子串 s_{i-x+1...i}
的散列值不等于长度为 x 的前缀的散列值，则返回 fase；否则返回 true
    {
        if((hash[i]-(hash[i-x]*cc)%mod+mod)%mod!=hash[x])
        {
            return false;
        }
    }
    return true;
}
int main()
{
    while(1)
```

```
    {
        scanf("%s",s+1);                    //输入字串
        len=strlen(s+1);                    //计算字串长度
        if(len==1 && s[1]=='.')             //返回空串的次幂 0
        {
            return 0;
        }
        for(int i=1;i<=len;i++)                 //计算所有前缀的散列值
        {
            hash[i]=(hash[i-1]*k+s[i])%mod;
        }
        for(int i=1;i<=len;i++)                 //枚举可能的子串长度
        {
            if(len%i==0 && check(i))     //若 s 能够划分出长度 i 的子串且所有相邻子串的
散列值相等，则输出子串个数，并退出 for 循环
            {
                printf("%d\n",len/i);
                break;
            }
        }
    }
}
```