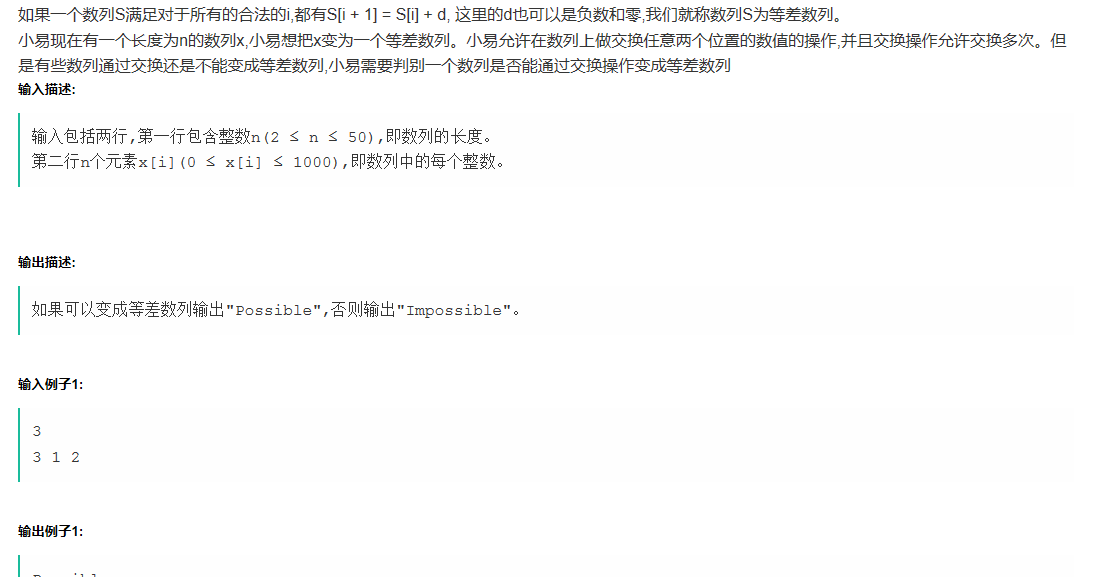
思路：就是看输入序列有多少种字符，若是大于2中，则肯定不存在只有一对不同颜色的相邻砖块

**def** Colored():  
 s = input()  
 **if** len(set(s)) == 2 :  
 print(2)  
 **elif** len(set(s)) == 1:  
 print(1)  
 **else**:  
 print(0)  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 Colored()



#include<iostream>

#include<cmath>

using namespace std;

int FindMin(int \*a,int n)

{

int i,min=a[0];

for(i=1;i<n-1;i++)

{

if(a[i]<min)

{

min = a[i];

}

}

return min;

}

int main()

{

int n,i,left=0,flag=0;

cin>>n;

int x[50],b[50];

for(i=0;i<n;i++)

{

cin>>x[i];

}

for(i=1;i<n;i++)

{

b[i-1] = abs(x[i] - x[left]);

}

int bmin = FindMin(b,n);

//cout<<bmin<<endl;

/\*for(i=0;i<n-1;i++)

{

cout<<b[i]<<endl;

}\*/

if(bmin <= 1)

{

int sum = 0;

for(i=0;i<n;i++)

{

sum = sum + x[i];

}

if(bmin == 1)

{

if(sum == (n+1)\*n/2)

{

cout<<"Possible"<<endl;

}

else

{

cout<<"Impossible"<<endl;

}

}

else

{

if(sum == n\*x[0])

{

cout<<"Possible"<<endl;

}

else

{

cout<<"Impossible"<<endl;

}

}

}

else

{

int j=0;

while(j<n-1)

{

if((b[j]%bmin) == 0)

{

// cout<<b[j]%bmin<<endl;

j += 1;

}

else

{

cout<<"Impossible"<<endl;

flag = 1;

break;

}

}

if(flag == 0)

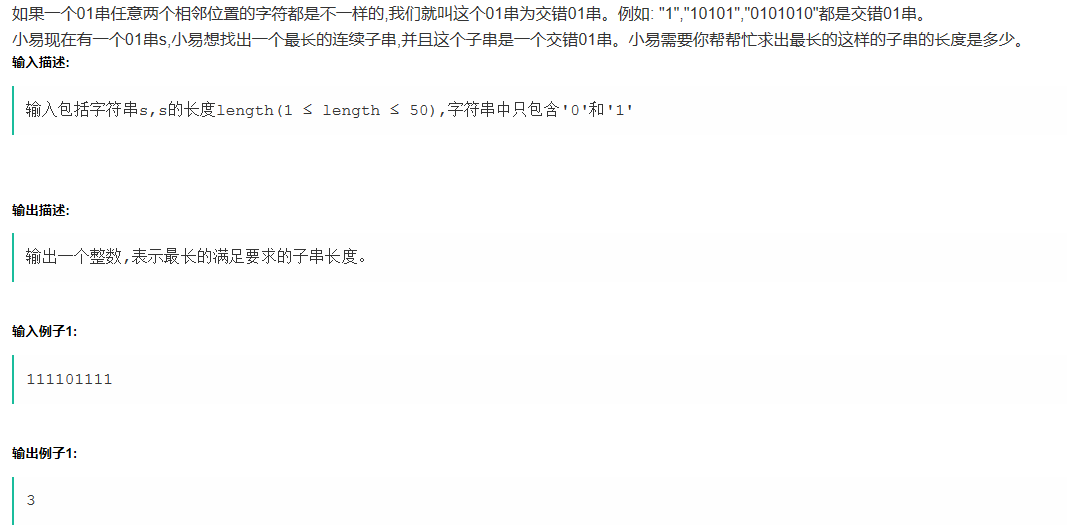
{

cout<<"Possible"<<endl;

}

}

}



#include<stdio.h>

int len(char \*a)

{

char \*y=a;

while(\*y++);

return (y-a-1);

}

int main()

{

char x[50];

gets(x);

int xlen = len(x);

// printf("%d\n",xlen);

int h[xlen],num[xlen];

int i,j,sum=0,cur=0,low,max=0,flag=0;

B:

low=cur;

A:

for(i=cur;i<xlen-1;i++)

{

j=i+1;

if(x[j]!=x[i])

{

sum++;

//printf("%d\n",sum);

cur=j;

goto A;

}

else

{

if(sum>max)

{

flag=1;

max=sum;

}

cur=low+1;

sum=0;

goto B;

}

}

if(flag==1)

{

printf("%d\n",max+1);

}

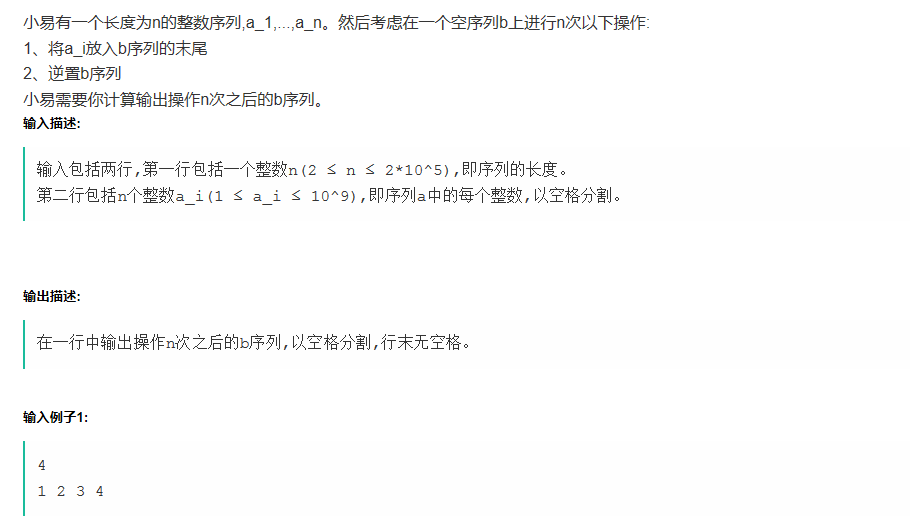
else

{

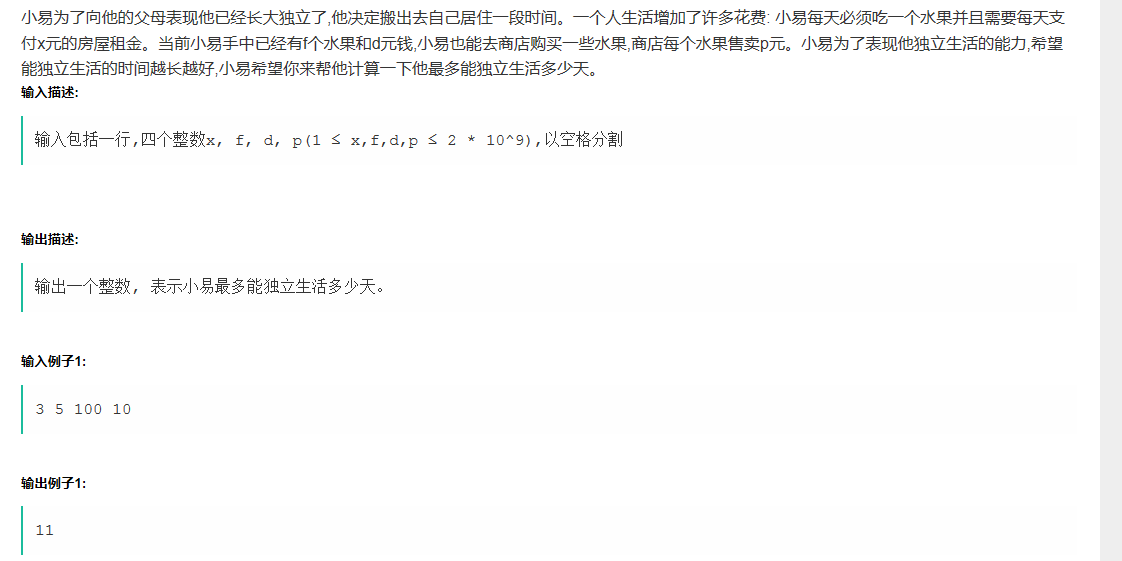
printf("%d\n",sum+1);

}

}

**from** collections **import** deque  
**def** Reverse(lists):  
 ReverseList = []  
 **for** i **in** range(0,len(lists)):  
 ReverseList.append(lists[len(lists)-i-1])  
 **return** ReverseList  
  
**def** OperationList(lists,n):  
 returnList = []  
 **for** i **in** range(0,n):  
 returnList.append(lists[i])  
 returnList = Reverse(returnList)  
 returnList = **''**.join(f+**' ' for** f **in** returnList)  
 print(returnList.strip())  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 OperatedNum = int(input())  
 lists = input().split(**" "**)  
 OperationList(lists,OperatedNum)



#include<iostream>

using namespace std;

int main()

{

int x,f,d,p,i,MaxNum;

cin>>x>>f>>d>>p;

int n = int(d/p);

for(i=0;i<n;i++)

{

if(f\*x>=d)

{

MaxNum = int(d/x);

}

else

{

if(x\*(f+i)>d-p\*i)

{

MaxNum = i+f-1;

break;

}

}

}

cout<<MaxNum<<endl;

}