2017校招全国统一模拟笔试第三场

1.牛牛想对一个数做若干次变换，直到这个数只剩下一位数字。  
变换的规则是：将这个数变成 所有位数上的数字的乘积。比如285经过一次变换后转化成2\*8\*5=80.  
问题是，要做多少次变换，使得这个数变成个位数。

#include<iostream>

using namespace std;

int main()

{

int n,temp,count = 0;

cin>>n;

while(n>9)

{

temp = 1;

while(n)

{

temp \*= n%10;

n=n/10;

}

n = temp;

count += 1;

}

cout<<count<<endl;

return 0;

}

2. 给出一个区间[a, b]，计算区间内“神奇数”的个数。  
神奇数的定义：存在不同位置的两个数位，组成一个两位数（且不含前导0），且这个两位数为质数。  
比如：153，可以使用数字3和数字1组成13，13是质数，满足神奇数。同样153可以找到31和53也为质数，只要找到一个质数即满足神奇数。

#include<iostream>

#include<math.h>

using namespace std;

bool NumIsPrime(int x)//判断是否为质数

{

for(int i=2;i<sqrt(x);i++)

{

if(x!=2&&x%i==0)

return false;

}

return true;

}

bool NumIsSq(int n)//判断是否为神奇数

{

int N,number[10];

for(N=0;n;N++)

{

number[N]=n%10;

n=n/10;

}

for(int i=0;i<N;i++)

{

for(int j=0;j<N;j++)

{

if(i==j)

{

continue;

}

if(number[i]!=0&&NumIsPrime(number[i]\*10+number[j]))

{

return true;

}

}

}

return false;

}

int main()

{

int a,b;

cin>>a>>b;

int n=0;

for(int i=a;i<=b;i++)

{

if(NumIsSq(i))

{

n++;

}

}

cout<<n<<endl;

return 0;

}

3. 牛牛手里有一个字符串A，羊羊的手里有一个字符串B，B的长度大于等于A，所以牛牛想把A串变得和B串一样长，这样羊羊就愿意和牛牛一起玩了。  
而且A的长度增加到和B串一样长的时候，对应的每一位相等的越多，羊羊就越喜欢。比如"abc"和"abd"对应相等的位数为2，为前两位。  
牛牛可以在A的开头或者结尾添加任意字符，使得长度和B一样。现在问牛牛对A串添加完字符之后，不相等的位数最少有多少位？

#include<iostream>

#include<string.h>

using namespace std;

int CmpStr(char \*strA,char \*strB)

{

int count=0,max=81;

int NA=strlen(strA),N=strlen(strB)-strlen(strA);

while(N>0)

{

for(int i=0;i<NA;i++)

{

if(strA[i]!=strB[i+N])

{

count++;

}

}

if(max>count)

{

max=count;

count=0;

}

N--;

}

return max;

}

int main()

{

char strA[81],strB[81];

cin>>strA>>strB;

cout<<CmpStr(strA,strB)<<endl;

return 0;

}

4.给定一个整数sum,从有N个有序元素的数组中寻找元素a,b,使得a+b的结果最接近sum

#include<iostream>

#include<cmath>

#include<climits>

using namespace std;

void FindCloserSum(int \*a,int n,int sum)

{

int i = 0,j = n - 1;

int Start = 0,End = 0;

int dif = INT\_MAX;//定义一个最大整数类型的最大值，应调用包#include<climits>

while(i < n && j > i)

{

if(abs(a[i]+a[j]-sum)<dif)

{

Start = a[i];

End = a[j];

dif = abs(a[i]+a[j]-sum);

}

if(a[i]+a[j]<sum)

{

i++;

}

else

{

j--;

}

}

cout<<Start<<" "<<End<<endl;

}

int main()

{

int a[] = {-5,1,0,1,3,4,5};

FindCloserSum(a,7,7);

return 0;

}

5. 在一个二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。杨氏矩阵

思路：记矩阵的右上角(左下角也可以)元素为a，搜索起点设置为a，要查找的元素为k：  
若a>k，则a所在列的所有元素均大于k，搜索位置左移1位，然后删除该列构成新的矩阵；  
若a<k，则a所在行的所有元素均小于k，搜索位置下移1位，然后删除该行构成新的矩阵；

#include<iostream>

using namespace std;

bool YangShiSearch(int a[4][4],int rows,int colnums,int number)

{

bool flag = false;

if(rows>0 && colnums>0)

{

int row=0,colnum=colnums-1;

while(row<rows && colnum>=0)

{

if(a[row][colnum] == number)

{

flag = true;

cout<<'('<<row<<','<<colnum<<')'<<endl;

break;

}

else if(a[row][colnum] > number)

{

colnum -= 1;

}

else

{

row += 1;

}

}

}

return flag;

}

int main()

{

int a[4][4]={{1,2,8,9},{2,4,9,12},{4,7,10,13},{6,8,11,15}};

int number;

cin>>number;

bool flag = YangShiSearch(a,4,4,number);

if(flag == false)

{

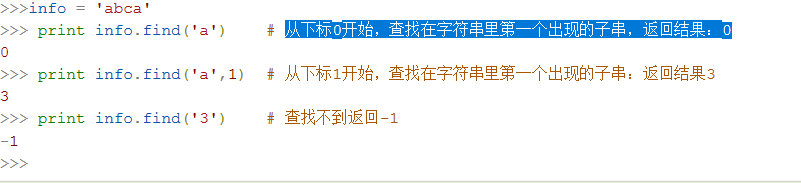
cout<<"没有"<<endl;

}

}

每次会显示两个字符串 s 和 t，根据古老的传说，牛牛需要每次都回答 t 是否是 s 的子序列。注意，子序列不要求在原字符串中是连续的，例如串 abc，它的子序列就有 {空串, a, b, c, ab, ac, bc, abc} 8 种。





x = input()

y = input()

t = 0

flag = 1

for j in y:

if x.find(j,t)==-1:// 从下标0开始，查找在字符串里第一个出现的子串，返回结果：0

flag=0

break

else:

t=x.find(j,t)+1

if flag==1:

print('Yes')

else:

print('No')



