Nowcode刷题

1、链接：<https://www.nowcoder.com/questionTerminal/18ecd0ecf5ef4fe9ba3f17f8d00d2d66?pos=11&orderByHotValue=1>  
来源：牛客网

Fibonacci数列是这样定义的：  
F[0] = 0  
F[1] = 1  
for each i ≥ 2: F[i] = F[i-1] + F[i-2]  
因此，Fibonacci数列就形如：0, 1, 1, 2, 3, 5, 8, 13, ...，在Fibonacci数列中的数我们称为Fibonacci数。给你一个N，你想让其变为一个Fibonacci数，每一步你可以把当前数字X变为X-1或者X+1，现在给你一个数N求最少需要多少步可以变为Fibonacci数。

输入描述:

输入为一个正整数N(1 ≤ N ≤ 1,000,000)

输出描述:

输出一个最小的步数变为Fibonacci数"

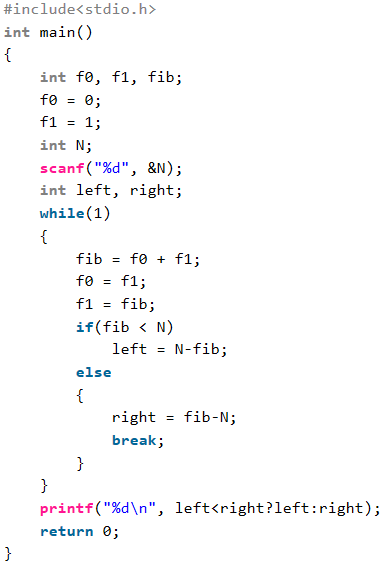
示例1

输入

15

输出

2



2、请实现一个函数，将一个字符串中的每个空格替换成“%20”。

例如，当字符串为We Are Happy.则经过替换之后的字符串为We%20Are%20Happy。

//采用临时空间

class Solution {

public:

void replaceSpace(char \*str,int length)

{

int n = 1000;

char \*tmp = (char\*)malloc(sizeof(char)\*n);

int i = 0;

char \*pstr = str;

while(\*pstr != '\0')

{

if(\*pstr == ' ')

{

tmp[i++] = '%';

tmp[i++] = '2';

tmp[i++] = '0';

}

else

tmp[i++] = \*pstr;

pstr++;

}

tmp[i] = '\0';

strcpy(str, tmp);

free(tmp);

}

};

//原字符串上进行修改

#include <iostream>

using namespace std;

//length为系统规定字符串输出的最大长度，固定为一个常数

class Solution {

public:

void replaceSpace(char \*str, int length){

int blankNumber = 0;//空格的数量

int oldstringLen;//记录原字符串的长度

//首先遍历原字符串，找出字符串的长度以及其中的空格数量

for (oldstringLen = 0; str[oldstringLen] != '\0'; oldstringLen++){

if (str[oldstringLen] == ' ')

blankNumber++;

}

//根据原字符串的长度和空格的数量我们可以求出最后新字符串的长度

int newstringLen = oldstringLen + blankNumber \* 2;//新字符串的长度

if (newstringLen>length)

return;

str[newstringLen] = '\0';//此行很重要，因为原字符串最后一个字符为'\0'

//设置两个指针point1和point2分别指向原字符串和新字符串的末尾位置

int point1 = oldstringLen - 1, point2 = newstringLen - 1;//因为'\0'已经手工加到最后新串的最后一个字符，所以减1咯

while (point1 >= 0 && point2>point1){//两指针相同时，跳出循环

if (str[point1] == ' '){//如果point1指向为空格，那么从point2开始赋值“02%”

str[point2--] = '0';

str[point2--] = '2';

str[point2--] = '%';

}

else //如果point1指向内容不为空格，那么将内容赋值给point2指向的位置

str[point2--] = str[point1];

point1--;//不管是if还是else都要把point1前移,为了下一次的执行

}

}

};

//C++解法

class Replacement

{

public:

string replaceSpace(string iniString, int length)

{

int pos;

while((pos = iniString.find(' ')) != iniString.npos)

{

iniString.replace(pos, 1, "%20");

}

return iniString;

}

};

3、题目描述

写出一个程序，接受一个由字母和数字组成的字符串，和一个字符，然后输出输入字符串中含有该字符的个数。不区分大小写。

输入描述:

第一行输入一个有字母和数字以及空格组成的字符串，第二行输入一个字符。

输出描述:

输出输入字符串中含有该字符的个数。

示例1

输入

ABCDEF

A

输出

1

#include<iostream>

#include<string>

using namespace std;

size\_t CalcCount(string &str, char ch)

{

size\_t count = 0;

for(int i=0; i<str.size(); ++i)

{

if(str[i]-ch==0 || abs(str[i]-ch)==32) //注意需要取绝对值

count++;

}

return count;

}

int main()

{

string str;

cin>>str;

char ch;

cin>>ch;

int count = CalcCount(str, ch);

cout<<count<<endl;

return 0;

}

4、题目描述

计算字符串最后一个单词的长度，单词以空格隔开。

https://www.nowcoder.com/practice/8c949ea5f36f422594b306a2300315da?tpId=37&tqId=21224&tPage=1&rp=&ru=%2Fta%2Fhuawei&qru=%2Fta%2Fhuawei%2Fquestion-ranking

输入描述:

一行字符串，非空，长度小于5000。

输出描述:

整数N，最后一个单词的长度。

示例1

输入

hello world

输出

5

#include<iostream>

using namespace std;

size\_t Count\_Last\_Word(string &str)

{

if(str.size() == 0)

return 0;

int count = 0;

int i = str.size()-1;

while(i>=0 && str[i] == ' ') //去掉尾部的空格

i--;

while(i>=0 && str[i]!=' ' && isalpha(str[i]))

{

count++;

i--;

}

return count;

}

int main()

{

string str;

//cin>>str;

getline(cin, str); //注意这个地方的输入，cin会以空格结束字符串，所以使用getline

cout<<Count\_Last\_Word(str)<<endl;

return 0;

}

5、题目描述

反转字符串

<https://leetcode-cn.com/problems/reverse-string>

编写一个函数，其作用是将输入的字符串反转过来。输入字符串以字符数组 char[] 的形式给出。

不要给另外的数组分配额外的空间，你必须原地修改输入数组、使用 O(1) 的额外空间解决这一问题。

你可以假设数组中的所有字符都是 ASCII 码表中的可打印字符。

示例 1：

输入：["h","e","l","l","o"]

输出：["o","l","l","e","h"]

示例 2：

输入：["H","a","n","n","a","h"]

输出：["h","a","n","n","a","H"]

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/reverse-string

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

class Solution

{

public:

void reverseString(vector<char>& s)

{

if(s.empty())

return;

string str;

for(int i=s.size()-1; i>=0; --i)

str.push\_back(s[i]);

for(int i=0; i<str.size(); ++i)

s[i] = str[i];

}

};

/\*

class Solution

{

public:

void reverseString(vector<char>& s)

{

if(s.empty())

return;

int start = 0;

int end = s.size()-1;

while(start < end)

{

swap(s[start], s[end]);

start++;

end--;

}

}

};

\*/

6、题目描述

给定一个字符串，找到它的第一个不重复的字符，并返回它的索引。如果不存在，则返回 -1。

案例:

s = "leetcode"

返回 0.

s = "loveleetcode",

返回 2.

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/first-unique-character-in-a-string

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

class Solution {

public:

    int firstUniqChar(string s)

    {

        if(s.empty())

            return -1;

        for(int i=0; i<s.size(); ++i)

        {

            int index = s.find(s[i]);

            int reverse\_index = s.rfind(s[i]);

            if(index == reverse\_index)

                return i;

        }

        return -1;

    }

};

7、输入一个字符串，求出该字符串包含的字符集合

<https://www.nowcoder.com/questionTerminal/784efd40ed8e465a84821c8f3970b7b5>

#include<iostream>

#include<string>

#include<string.h>

using namespace std;

string UniqueStr(string &str)

{

char hash[123] = {0};

string tmp;

for(int i=0; i<str.size(); ++i)

{

if(hash[str[i]] == 0)

{

hash[str[i]]++;

tmp.push\_back(str[i]);

}

}

return tmp;

}

int main()

{

string str, res;

while(cin>>str)

{

res = UniqueStr(str);

cout<<res<<endl;

res.clear();

}

return 0;

}

/\*

int main()

{

char hash[123] = {0};

string str;

while(cin>>str)

{

for(int i=0; i<str.size(); ++i)

{

if(hash[str[i]] == 0)

{

hash[str[i]]++;

//printf("%c", str[i]);

cout<<str[i];

}

}

//printf("\n");

cout<<endl;

memset(hash,0,123);

}

return 0;

}

\*/

8、链接：<https://www.nowcoder.com/questionTerminal/72ff6503455c4a008675e79247ef2a3a>  
来源：牛客网

有一个排过序的数组，包含n个整数，但是这个数组向左进行了一定长度的移位，例如，原数组为[1,2,3,4,5,6]，向左移位5个位置即变成了[6,1,2,3,4,5],现在对于移位后的数组，需要查找某个元素的位置。请设计一个复杂度为log级别的算法完成这个任务。

给定一个int数组**A**，为移位后的数组，同时给定数组大小**n**和需要查找的元素的值**x**，请返回x的位置(位置从零开始)。保证数组中元素互异。

测试样例：

[6,1,2,3,4,5],6,6

返回：0

class Finder {

public:

int findElement(vector<int> A, int n, int x) {

// write code here

int left, right, mid;

left = 0; right = n-1;

while(left <= right)

{

mid = (left+right)/2;

if(x == A[mid])

return mid;

if(x > A[mid])

{

if(A[left]>A[mid] && x>A[right])

right = mid-1;

else

left = mid+1;

}

else

{

//if(A[right]>x && A[left]<A[mid])

if(A[mid]>A[left] && x<A[left])

{

left = mid+1;

}

else

{

right = mid-1;

}

}

}

return -1;

}

};