C++方向编程题答案

答案说明：

> 大家如果对本次题目或者答案有问题，可以联系下方的出题老师答疑。
>
> **出题老师：**
>
> 选择题：鲍松山 qq：365690203
>
> 代码题1：鲍松山 qq：365690203
>
> 代码题2：张文超 qq：3627274478

day17

> **题目ID：36877-杨辉三角的变形**
>
> 链接：https://www.nowcoder.com/practice/8ef655edf42d4e08b44be4d777edbf43?tpId=37&&tqId=21276&rp=1&ru=/activity/oj&qru=/ta/huawei/question-ranking
>
> **【解题思路】：**
>
> 按照题目意思，可以发现第n行有2n - 1个元素，第i,j元素等于上一行第j - 2,j - 1,j三列元素之和，每一行的第一列和最后一列都为1，如果是第二列，则只是两个元素之和。
>
> **【示例代码】**

```
#include<iostream>
#include<string>
#include<vector>

using namespace std;
int main()
{
    int n, m;
    while (cin >> n)
    {
        m = 2 * n - 1;
        vector<vector<int>> dp(n, vector<int>(m, 0));
        dp[0][0] = 1;
        for (int i = 1; i < n; i++)
        {
//第一列和最后一列都为1
            dp[i][0] = dp[i][2 * i] = 1;
            for (int j = 1; j < 2 * i; ++j)
            {
                if (j == 1)
        //如果是第二列，则只是两个元素之和
                    dp[i][j] = dp[i - 1][j - 1] + dp[i - 1][j];
                else
        //第i,j元素等于上一行第j - 2,j - 1,j三列元素之和

                    dp[i][j] = dp[i - 1][j - 2] + dp[i - 1][j - 1] + dp[i - 1][j];
```

```
            }
        }
        int k;
        for (k = 0; k < m; k++)
        {
            if (dp[n - 1][k] % 2 == 0 && dp[n - 1][k] != 0)
            {
                cout << k + 1 << endl;
                break;
            }
        }
        if (k == m)
            cout << -1 << endl;
    }
    return 0;
}
```

> **【更正】：**

由于牛客更新了测试用例，加强了内存限制，因此上述用例在运行时会超出内存限制，所以上述使用生成杨辉三角，然后在第n行上寻找偶数的方式已经不合适，故进行更正。

通过以上的数据分析规律为 {-1,-1,2,3,2,4,2,3,2,4,...}，即第n行上偶数所在的下标，具体分析可见录屏。

【示例代码】

```
#include<iostream>
#include<string>
using namespace std;
int main(int argc,char* argv[])
{
 int nRow=0;
 while(cin>>nRow)
 {
     int res=-1;
     int myInt[]={4,2,3,2};
     if(nRow>2)
         res=myInt[(nRow-2)%4];
     cout<<res<<endl;
 }
 return 0;
}
```

题号：36826 计算某字符出现次数

【链接】

https://www.nowcoder.com/questionTerminal/a35ce98431874e3a820dbe4b2d0508b1

【题目描述】

写出一个程序，接受一个由字母、数字和空格组成的字符串，和一个字符，然后输出输入字符串中该字符的出现次数。（不区分大小写字母）

数据范围： `1 ≤ n ≤ 1000`

【题目解析】

首先，字符串中包含有空格，因此我们使用`gets`函数获取一行数据，而不用`scanf`，因为`scanf`函数会默认以空格截断输入作为输入的结束.

其次，因为在计算字符出现次数中，要求是不区分大小写的，因此在处理过程中，如果输入的是小写字母，则计数不但要加上小写字母的计数，还要加上大写字母的计数，反之大写字母也是同样如此，而在`ascii`表中，大写字母和小写字母的差值是`32` （`'a'的ascii值是97`;`'A'的ascii值是65`）

最后，实现思路，因为`ascii字符`只有`128`个，他们的实际存储是`0~127`，那么我们只需要定义一个具有`128`个整形元素的数组，第`0`号下标位置存放`0`对应的`ascii字符`出现次数，`1`号下标位置存放`1`对应的`ascii字符`的次数....以此类推，数组每个位置存放的就是对应`ascii字符`出现的次数。最终以指定字符为下标获取它出现的次数进行打印。

【答案】

```
#include <stdio.h>
#include <string.h>

int main()
{
    char str[1003] = { 0 };
    //gets函数获取一行数据，因为会连带最后的换行一起获取，所以不用担心遗留的换行对后续输入造成影响
    gets(str);
    char find;
    scanf("%c", &find);//捕捉要查找的字符
    int count[128] = { 0 };//用于记录每个字符出现次数
    int len = strlen(str);//获取字符串实际长度
    for (int i = 0; i < len; i++) {
        count[str[i]] += 1; //对应字符的位置计数+1
    }
    int res = count[find];
    if (find >= 'A' && find <= 'Z') {//如果是A-Z需要将a-z对应的计数加上
        res += count[find + 32];
    }
    else if (find >= 'a' && find <= 'z') {//如果是a-z需要将A-Z对应的计数加上
        res += count[find - 32];
    }
    printf("%d\n", res);
    return 0;
}
```


题目ID:23270-二叉树的镜像【作废】

链接：https://www.nowcoder.com/questionTerminal/a9d0ecbacef9410ca97463e4a5c83be7

【题目解析】

无

【解题思路】

源二叉树：

```
        8
       /  \
      6   10
     / \  / \
    5  7 9 11
```

镜像二叉树：

```
        8
       /  \
      10   6
     / \  / \
    11 9 7   5
```

通过上述图示，可以看出，源二叉树镜面成像变成镜像二叉树。 一个节点的左右节点进行互换，可以通过递归来实现。

【示例代码】

```
class Solution {
public:
    void Mirror(TreeNode *pRoot) {

        if(pRoot == NULL)
           return pRoot;
        //节点的左右子节点为null(即就是节点为叶子节点) 同样不处理
        if(pRoot->left == NULL && pRoot->right == NULL)
           return pRoot;
        //节点的左右子节点交换
        TreeNode *pTemp = pRoot->left;
        pRoot->left = pRoot->right;
        pRoot->right = pTemp;
        //递归处理
        if(pRoot->left != NULL)
           Mirror(pRoot->left);
        if(pRoot->right != NULL)
           Mirror(pRoot->right);
           return pRoot;
        }
};
```