问题 A: 递归实现逆序输出（递归函数）

时间限制: 1 Sec 内存限制: 128 MB
提交: 975 解决: 694
[提交][状态][讨论版]

题目描述

利用**递归函数调用**方式，将所输入的任意个字符，以相反顺序打印出来。

输入

字符串长度和该字符串

输出

逆序输出

样例输入

```
5
12345
```

样例输出

```
54321
```

```
#include <stdio.h>

void reverseOutput(char a[100], int len){
        if (len == 0)
                return ;
        printf("%c", a[len-1]);
        reverseOutput(a, len-1);
}

int main(){
        int len;
        scanf("%d", &len);
        char a[100];
        getchar();
        scanf("%s", a);
        reverseOutput(a, len);

        return 0;
}
```

问题 B: 递归实现斐波那契数列

时间限制: 1 Sec 内存限制: 128 MB
提交: 169 解决: 149
[提交][状态][讨论版]

题目描述

斐波那契数列的定义如下:

也就是说，斐波那契数列由1 和1开始，之后的每一项是之前的两数相加，例如:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233,...

要求**用递归函数**实现，求斐波那契数列第n项的值。

输入

测试次数和每次要计算的项数

输出

计算的结果

样例输入

```
5
1
2
5
10
20
```

样例输出

```
1
1
5
55
6765
```

```
#include <stdio.h>

int Fib(int num){
    if (num == 1 || num == 2){
        return 1;
    }
    else{
        return Fib(num - 1) + Fib(num - 2);
```

```
    }
}

int main(){
    int T;
    scanf("%d", &T);
    while (T --){
        int n;
        scanf("%d", &n);
        printf("%d\n", Fib(n));
    }

    return 0;
}
```

问题 C: 整数转字符串（递归）

时间限制: 1 Sec 内存限制: 128 MB
提交: 1513 解决: 715
[提交][状态][讨论版]

题目描述

写递归函数void itostr(int num,char str[])，函数功能是将一个整数num转换为字符串str，如整数135，转换为字符串“135”。主函数如下，不可修改。

itostr每次独立调用。不可以为实现itostr的递归定义全局变量。

```
void itostr(int num,char str[])
int main()
{
    const int SIZE = 20;
    int t,num;
    char str[SIZE];

    cin>>t;
    while(t--)
    {
        cin>>num;
        itostr(num,str);
        cout<<str<<endl;
    }
    return 0;
}
```

输入

测试次数t

t个整数

输出

每个整数，输出转换后的数字字符串

样例输入

```
5
135
0
78934012
-1110
1323
```

样例输出

```
135
0
78934012
-1110
1323
```

```
#include <stdlib.h>
#include <cmath>
#include <iostream>
using namespace std;
void itostr(int num,char str[]){
        if(num==0){
                str[0]='0';
                str[1]=0;
                return ;
        }
        if(num<0){
                str[0]='-';
                str[int(log10(-num))+2]=0;
                if(num/10!=0)itostr(num/10,str);
                str[int(log10(-num))+1]=abs(num%10)+'0';
        }
        if(num>0){
                str[int(log10(num)+1)]=0;
                if(num/10)itostr(num/10,str);
                str[int(log10(num))]=num%10+'0';
        }
}
int main()
{
        const int SIZE=20;
        int t,num;
        char str[SIZE];
```

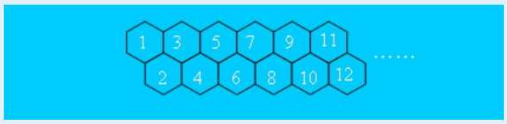
```
        cin>>t;
        while(t--)
        {
                cin>>num;
                itostr(num,str);
                cout<<str<<endl;
        }
        return 0;
}
```

问题 D: 一只小蜜蜂（递归）

时间限制: 1 Sec 内存限制: 128 MB
提交: 1104 解决: 721
[提交][状态][讨论版]

题目描述

有一只经过训练的蜜蜂只能爬向右侧相邻的蜂房，不能反向爬行。请编程计算蜜蜂从蜂房a爬到蜂房b的可能路线数。其中，蜂房的结构如下所示。



提示：定义函数int GetPathNumber(int a,int b)计算从a到b的路线数并返回。找出它和a+1到b，a+2到b的路线数关系。

输入

输入数据的第一行是一个整数N,表示测试实例的个数，然后是N 行数据，每行包含两个整数a和b(0<a<b<50)。

输出

对于每个测试实例，请输出蜜蜂从蜂房a爬到蜂房b的可能路线数，每个实例的输出占一行。

样例输入

```
2
1 2
3 6
```

样例输出

```
1
3
```

```
#include <stdio.h>

int GetPathNumber(int a, int b){
        if (b - a == 1)
                return 1;
        else if (b - a == 2)
                return 2;
        else
```

```
                return GetPathNumber(a+1, b) + GetPathNumber(a+2,
b);
}

int main(){
        int T;
        scanf("%d", &T);
        while (T --){
                int a, b;
                scanf("%d %d", &a, &b);
                printf("%d\n", GetPathNumber(a, b));
        }
        return 0;
}
```

问题 U: 6174问题

时间限制: 1 Sec 内存限制: 128 MB
提交: 1232 解决: 521
[提交][状态][讨论版]

题目描述

假设你有一个各位数字互不相同的四位数，把所有的数字从大到小排序后得到a,从小到大后得到b,然后用a-b替换原来这个数，并且继续操作。例如，从1234出发，依次可以得到4321-1234=3087、8730-378=8352、8532-2358=6174，又回到了它自己！现在要你写一个程序来判断一个四位数经过多少次这样的操作能出现循环，并且求出操作的次数

比如输入1234执行顺序是1234->3087->8352->6174->6174,输出是4。

要求编写函数int circle(int num)，求数字num按上述规则运算，出现循环数据的操作次数，并返回计算结果。

输入

第一行输入n,代表有n组测试数据。
接下来n行每行都写一个各位数字互不相同的四位数

输出

经过多少次上面描述的操作才能出现循环

样例输入

```
1
1234
```

样例输出

```
4
```

```
#include <stdio.h>

int times;
int circle(int num){
        int arr[4] = {num/1000, (num/100)%10, (num/10)%10, num%10};
        int temp;
        for (int i = 3; i > 0; i --){
                for (int j = 0; j < i; j ++){
                        if (arr[i] > arr[j]){
```

```
                                temp = arr[i];
                                arr[i] = arr[j];
                                arr[j] = temp;
                        }
                }
        }
        int maxNum = arr[0] * 1000 + arr[1] * 100 + arr[2] * 10 +
arr[3];
        for (int i = 3; i > 0; i --){
                for (int j = 0; j < i; j ++){
                        if (arr[i] < arr[j]){
                                temp = arr[i];
                                arr[i] = arr[j];
                                arr[j] = temp;
                        }
                }
        }
        int minNum = arr[0] * 1000 + arr[1] * 100 + arr[2] * 10 +
arr[3];
        times += 1;
        if (maxNum - minNum == num){
                return times;
        }
        else{
                return circle(maxNum - minNum);
        }
}

int main(){
        int T;
        scanf("%d", &T);
        while (T --){
                int n;
                scanf("%d", &n);
                times = 0;
                printf("%d\n", circle(n));
        }
        return 0;
}
```

问题 V: 盗梦空间

时间限制: 1 Sec 内存限制: 128 MB
提交: 1944 解决: 562
[提交][状态][讨论版]

题目描述

《盗梦空间》是一部精彩的影片，在这部电影里，Cobb等人可以进入梦境之中，梦境里的时间会比现实中的时间过得快得多，这里假设现实中的3分钟，在梦里就是1小时。

然而，Cobb他们利用强效镇静剂，可以从第一层梦境进入第二层梦境，甚至进入三层，四层梦境，每层梦境都会产生同样的时间加速效果。那么现在给你Cobb在各层梦境中经历的时间，你能算出现实世界过了多长时间吗?

比如，Cobb先在第一层梦境待了1个小时，又在第二层梦境里待了1天，之后，返回第一层梦境之后立刻返回了现实。

那么在现实世界里，其实过了396秒（6.6分钟）

输入

第一行输入一个整数T（0<=T<=100），表示测试数据的组数。

每组测试数据的第一行是一个数字M(3<=M<=100)
随后的M行每行的开头是一个字符串，该字符串如果是"IN" 则Cobb向更深层的梦境出发了，如果是字符串"OUT"则表示Cobb从深层的梦回到了上一层。如果是首字符串是"STAY"则表示Cobb在该层梦境中停留了一段时间，本行随后将是一个整数S表示在该层停留了S分钟（1<=S<=10000000)。数据保证在现实世界中，时间过了整数秒。

输出

对于每组测试数据，输出现实世界过的时间（以秒为单位）。

样例输入

```
1
6
IN
STAY 60
IN
STAY 1440
OUT
OUT
```

样例输出

```
396
```

```
#include <stdio.h>

char action[5];
int dreaming(int move, int times, int sign){
        if (move == 0){
                return times;
        }
        scanf("%s", action);
        if (action[0] == 'I'){
                return dreaming(move-1, times, sign*20);
        }
        else if (action[0] == 'O'){
                return dreaming(move-1, times, sign/20);
        }
        else if (action[0] == 'S'){
                int second;
                scanf("%d", &second);
```

```
                times += (second*60) / sign;
                return dreaming(move-1, times, sign);
        }
}

int main(){
        int T;
        scanf("%d", &T);
        while (T --){
                int n;
                scanf("%d", &n);
                printf("%d\n", dreaming(n, 0, 1));
        }
        return 0;
}
```