A 猴子吃桃问题

时间限制：1000ms　内存限制：65536kb

通过率：1700/1740 (97.70%)　正确率：1700/2624 (64.79%)

题目描述

有一堆桃子不知数目，猴子第一天晚上吃掉一半，又多吃了一个，第二天照此方法，吃掉剩下桃子的一半又多一个，天天如此，到第 m 天早上，猴子发现只剩一只桃子了，问这堆桃子原来有多少个？

输入

第一行为一个整数 n,表示有 n 组测试数据,
从第二行开始，每一行的数据为第 m 天(1<m<29)

输出

n 行，每行一个数，代表桃子的总个数

输入样例

```
2
2
3
```

输出样例

```
4
10
```

考察知识点

循环
难度系数： 1

解题思路

这道题是送分题，学过了循环就能做了。不过要稍微注意一下循环的终止条件，题目说的是第 m 天早上发现只剩一个桃子，那么其实猴子只吃了 m-1 天的桃子，要计算桃子总数倒着算回去算到 m-1 天就可以了
##

参考代码

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

int main()
{
    int T;
    int m;
    int i;
    int result;

    scanf("%d",&T);

    while(T--){
        scanf("%d",&m);

        result=1;
        for(i=1;i<m;i++)
            result=(result+1)*2;

        printf("%d\n",result);
    }
    return 0;
}
```

B 傻傻 Aqi 的巧克力豆冰激凌

时间限制：1000ms　内存限制：65536kb

通过率：1185/1344 (88.17%)　正确率：1185/3007 (39.41%)

题目描述

傻傻 Aqi 带 Alice 出去玩，天气好热呀，傻傻 Aqi 想吃一种上面撒着巧克力豆的冰激凌，

其上巧克力豆的数量是可选的。冰激凌款式共有 nn 款 ，每款上面的巧克力豆数量可能不同、也可能相同。傻傻 Aqi 自然想吃巧克力豆多的，可 Alice 觉得巧克力吃多了对牙齿不好，最后两人决定买巧克力豆第三多的那款~那么，所以傻傻 Aqi 可以吃到多少颗巧克力豆呢？

输入一个正整数 nn，然后输入 nn 个正整数，nn 个正整数中可能有重复值。如果其中存在第 3 大的正整数，输出该数；否则，输出 0。

输入

单组数据输入,

第一行输入 1 个正整数 n（n≤50）n（n≤50），表示共有 n 款,

第二行输入 n 个正整数，表示每款上面巧克力豆的数量 x（x≤200）x（x≤200）。

输出

对于该组输入，
如果存在巧克力豆第三多的冰激凌，输出其巧克力豆数量是多少；否则输出 0。

输入样例 1

```
10
100 110 120 130 140 150 50 40 30 20
```

输出样例 1

```
130
```

输入样例 2

```
5
10 10 5 5 15
```

输出样例 2

```
5
```

考察知识点

```
排序
难度系数: 3
```

解题思路

这题实际是考察排序，底下的代码使用快速排序 qsort 函数从大到小排序解决，当然其他的排序方法也是可以的。然后找到排序后第三大的数字，注意其中可能有重复值，以下代码思路是将其中的重复元素去掉后的元素组成一个新数组，然后输出第 3 个元素，即为所要值。或者直接在从最大的值到最小值按顺序遍历的过程中，记录不同数据的个数，当出现第 3 个不同值时输出也可。

代码

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 51

int cmp(const void *a,const void *b){
    return *(int*)b-*(int*)a;
}

int main(){
    int N,cost[MAX],i,j,costfinal[MAX];
    scanf("%d",&N);
```

```
    for(i=0;i<N;i++){
        scanf("%d",&cost[i]);
    }
    qsort(cost,N,sizeof(cost[0]),cmp);
    j=1;
    costfinal[0]=cost[0];
    for(i=1;i<N;i++){
        if(cost[i]!=cost[i-1]){
            costfinal[j]=cost[i];
            j++;
        }
    }
    if(j>=3){
        printf("%d",costfinal[2]);
    }else{
        printf("0");
    }

    return 0;
}
```

C 位图

时间限制：1000ms　内存限制：65536kb

通过率：923/1063 (86.83%)　正确率：923/2186 (42.22%)

题目描述

给出一个大小为 n 行*m 列的矩形位图。该位图的每一个象素点不是白色就是黑色，但是至少有一个象素点是白色。在 i 行 j 列的象素点我们称为点（i，j）。

两个象素点 p1=(i1,j1)和 p2=(i2,j2)之间的距离定义如下：

$d(p1,p2)=|i1-i2|+|j1-j2|$。

现在的任务是：对于每一个象素点，计算它到最近的白色点的距离。如果它本身是白色点，距离为 0。

输入

第 1 行：4 个整数 n,m,x,y (用空格分割)（1<=n<=20,1<=m<=20）
接下来 n*m 个整数,分别是代表(1,1) (1,2) .... (2,1) (2,2)...以此类推
如果点（i，j）为白色，则值为 1，否则值为 0。

输出

求像素点(x,y),计算它到最近的白色点的距离。如果它本身是白色点，距离为 0。
输出一行,一个整数,表示距离。

输入样例

```
3 4 1 1
0 0 0 1
0 0 1 1
0 1 1 0
```

输出样例

```
3
```

考察知识点

```
二维数组 枚举
难度系数： 3
```

思路解析:

用二维数组输入,再用二重循环,进行简单的判断,然后用宏或者函数 abs 进行距离比较然后更新答案.

题目代码:

```
#include<stdio.h>
#define Abs(x) ((x) > 0 ? (x) :(-(x)))
```

```
#define MAXN 200
int a[MAXN +10][MAXN +10];
int main()
{
    int i,j,x,y,n,m;
    scanf("%d%d%d%d",&n,&m,&x,&y);
    int ans = n*m;
    for(i = 1;i <= n;i++)
        for(j = 1;j <= m;j++)
            scanf("%d",&a[i][j]);
    for(i = 1;i <= n;i++)
        for(j = 1;j <= m;j++)
            if(a[i][j] == 1 && ans > (Abs(x-i) + Abs(y-j)))
                ans = Abs(x-i) + Abs(y-j);
    printf("%d\n",ans);
    return 0;
}
```

D Time Conversion

时间限制: 1000 ms 内存限制: 65536 kb
总通过人数: 906 总提交人数: 1091

题目描述

给定 12 小时 AM / PM 格式的时间，将其转换为军事（24 小时）时间。
注意：午夜时间为 12 小时的 12:00:00AM，24 小时的 00:00:00。中午为 12 小时的 12:00:00PM，24 小时的 12:00:00。

输入

一个包含时间格式为 12 小时制的字符串（即：hh:mm:ssAM 或 hh:mm:ssPM），其中 01<=hh<=12 和 00<=mm,ss<=59。

输出

转换并以 24 小时格式输出给定的时间（即：hh:mm:ss），其中 00<=hh<=23。

输入样例

```
07:05:45PM
```

输出样例

```
19:05:45
```

考察知识点：

字符串处理

解题思路：

观察到读入的字符串只需要修改前两位并且删除后两位。删除操作我们可以将倒数第二位置为'\0'。修改分三种情况，一是中午 12 点，不需要修改；午夜 12 点，前两位修改为 00；其他情况，第一位加 1，第二位加 2。

代码：

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

void timeConversion(char* s) {
    int len = strlen(s);
    s[len - 1] = '\0';
    if (s[len - 2] == 'A') {
        if (s[0] == '1' && s[1] == '2') {
            s[0] = '0';
            s[1] = '0';
        }
    } else {
        if (s[0] != '1' || s[1] != '2') {
            s[0] += 1;
            s[1] += 2;
        }
    }
    s[len - 2] = '\0';
```

```
}

int main() {
    char s[20];
    scanf("%s", s);
    timeConversion(s);
    printf("%s", s);
    return 0;
}
```

E 质数的个数

时间限制：1000ms　内存限制：65536kb

通过率：965/1136 (84.95%)　正确率：965/3305 (29.20%)

题目描述

给出两个非负数，求出两个数之间的质数的个数。

输入

两个 short 型非负数 **a,b**

输出

不小于 **a** 且不大于 **b** 的所有质数的个数

输入样例

```
2 5
```

输出样例

```
3
```

考察知识点

> 质数
> 难度系数： 3

题解：

判断整个区间的质数个数,具体实现可以参看课件，可以分别判小于上限的个数和小于下限的个数，也可以在上限判断中记录结果，注意当范围较小时需要特判。

代码

```
#include<stdio.h>
int anw_z[] = {0,0,1,2,2,3,3,4};
int isPrime(int primes[], int m)
{
    if(m<2) return 0;
    int i;
    for(i=0; primes[i]*primes[i] <= m; i++)
    {
        if (m % primes[i] == 0)
            return 0;
    }
    return 1;
}
int init(int primes[],int Q, int P) //要求 N >= 3
{
    primes[0] = 2; primes[1] = 3; primes[2] = 5;
    int i, j, count, num, gap,anw=0;
    if(P<7) return anw_z[P] - anw_z[Q] + isPrime(primes, Q);
    if(Q<7) anw = anw_z[6] - anw_z[Q] + isPrime(primes, Q);
     //头 3 个质数直接给
    count = 3; num = 7; gap = 2;
    while(num <= P)
    {
        gap = 6 - gap; // 构造 2-4-2-4-...序列，减少遍历
        if (isPrime(primes, num))
        {
            primes[count++] = num;
```

```
            if(num>=Q) anw++;
        }
        num += gap;

    }
    return anw;
}
int p[100000];
int main()
{
    int x,y;
    scanf("%d%d",&x,&y);
    printf("%d",init(p,x,y));
}
```

F Terry 与哥德巴赫猜想

时间限制：1000ms　内存限制：65536kb

通过率：670/814 (82.31%)　正确率：670/1597 (41.95%)

题目描述

请你编一个程序验证哥德巴赫猜想。

先给出一个奇数 n，要求输出 3 个质数，这 3 个质数之和等于输入的奇数。

输入

一行，一个奇数 n，其中 9<n<20000。

输出

仅有一行，输出 3 个质数，这 3 个质数之和等于输入的奇数。相邻两个质数之间用一个空格隔开，最后一个质数后面没有空格。如果表示方法不唯一，请输出第一个质数最小的方案，如果第一个质数最小的方案不唯一，请输出第一个质数最小的同时，第二个质数最小的方案。

按从小到大的顺序输出

输入样例

```
2009
```

输出样例

```
3 3 2003
```

考察知识点

```
枚举
难度系数：3
```

解题思路

本题要求给一个奇数，求出三个质数，使这三个质数之和等于这个奇数。
那么我们想到可以枚举前两个数，第三个数等于所给奇数减去前两个数。如果这三个数都是质数，那么就满足条件。另外要求从小到大顺序输出，那么枚举的时候从小到大进行枚举。
为了方便判定，将判断质数的代码包装为一个函数 prime。具体操作见代码。

标程

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

int prime(int x)
{
    int i;
    for (i=2;i<=sqrt(x);i++)
        if (x%i==0)
```

```
            return 1;
    return 0;
}
int main()
{
    int a,n1,n2,n3,x;
    scanf("%d",&a);
    for (n1=2;n1<=a-4;n1++)
        for (n2=2;n2<=a-4;n2++)
        {
            n3=a-n1-n2;
            if (prime(n1)+prime(n2)+prime(n3)!=0)
                continue;
            printf("%d %d %d",n1,n2,n3);
            return 0;
        }
}
```

G Cut the sticks

时间限制: 1000 ms 内存限制: 65536 kb

总通过人数: 544 总提交人数: 683

题目描述

给你 N 根长度不相等的木棍。有一种切割操作：把所有的木棍都切去最短的高度，并记录下切下来的宽度（木棍数），扔掉切下的部分。重复这种切割操作，直到所有的木棍都被扔掉。 你需要输出每次切割下记录的宽度（木棍数）。

输入

第一行包含 N，下一行包含个空格分隔的数 A*i*。*(1 <=N <= 1000, 1*<=Ai<=1000)

输出

输出每次操作切下的木棍宽度。

输入样例 1

```
8
1 2 3 4 3 3 2 1
```

输出样例 1

```
8
6
4
1
```

输入样例 2

```
3
2 2 2
```

输出样例 2

```
3
```

考察知识点：

数学

解题思路一：

直观的想法，将所有木棍长度用数组存储，每次输出当前长度非 0 的木棍个数，并且长度减小。

代码：

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

```
#define N (1 << 10)
int arr[N];
int main() {
    int n, i;
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int size = n;
    while (size != 0) {
        printf("%d\n", size);
        int min = 1 << 10;
        for (i = 0; i < n; i++)
            if (arr[i] != 0 && arr[i] < min)
                min = arr[i];

        for (i = 0; i < n; i++)
            size -= (arr[i] != 0 && (arr[i] = arr[i] - min) == 0);
    }
    return 0;
}
```

解题思路二：

观察到，每次切割都会割掉所有长度最小的木棍，所以我们可以每次减去当前长度最小的木棍个数后输出。

代码：

```
#include<stdio.h>
#define N (1 << 10)
int arr[N];
int main() {
    int n, i, len;
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%d", &len);
        arr[len]++;
    }
    for (i = 1; i <= 1000; i++)
        if (arr[i]) {
            printf("%d\n", n);
            n -= arr[i];
```

```
        }
    return 0;
}
```

H Tarpe 酋长的砥砺

时间限制：1000ms　内存限制：65536kb

通过率：577/673 (85.74%)　正确率：577/1186 (48.65%)

题目描述

酋长每隔一段时间会给自己加一次 buff（爆肝状态）来对付过多的 ddl，但是酋长的 buff 每次只能持续 x 天，之后又会陷入长时间摸鱼状态，每学期初酋长会给自己制订计划（开启 buff 的时间点），请你计算一下酋长一学期一共需要爆肝多少天。
已经处于 buff 状态时如果再次开启状态，则 buff 会相应延长

输入

第一个数 n (0<n<1000)
表示开启 buff 的时间点个数，
接下来 n 个数构成一个上升序列，表示开启 buff 的时间点。(在 int 范围内)
接下来一个数 x，表示每次 buff 持续的天数。(在 int 范围内）

输出

一行，爆肝的总天数。

输入样例 1

```
2
1 4
2
```

输出样例 1

```
4
```

输入样例 2

```
2
1 2
2
```

输出样例 2

```
3
```

样例解释

在第一个样例中，第一次 buff 在第一天开启，持续到第二天结束，持续了两天。

第二次 buff 在第4天开启，持续到第5天结束，持续了两天。一共是4天。

在第二个样例中，第一次 buff 在第一天开启，而第二天仍处在 buff 状态，再次开启 buff 使得 buff 延长到第3天结束，一共持续了3天。

考察知识点

数组

难度系数：4

解题思路：

由于给的时间点是递增的，所以我们可以以每个时间点为对象来考虑，当碰到下一个开启的时间点时，我们先判断该书间点是否处于 buff 状态，如果处于 buff 状态，说明上一次 buff 还没结束，那么只要将该时间点减去上一个时间点就可以得到这段时间的长度，如果不处于 buff 状态，那么说明两个时间点之间 buff 的持续状态就是一个持续长度。

坑点：

最后一次持续期要记得加上。

AC 代码：

```
#include<stdio.h>
#include<stdlib.h>
#define MAXN 1005
int timeSeries[MAXN];
int n;
int duration;

int main()
{
    //freopen("in3.txt","r",stdin);
    //freopen("out3.txt","w",stdout);
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%d",&timeSeries[i]);
    }
    scanf("%d",&duration);
    int sum = 0;
    int i = 0;
    while(i<n-1)
    {
        sum                                                                 +=
(timeSeries[i]+duration>=timeSeries[i+1]?timeSeries[i+1]-
timeSeries[i]:duration);
        i++;
    }
```

```
    sum += duration;
    printf("%d\n",sum);
    //fclose(stdin);
    //fclose(stdout);
}
```

I 比赛排行榜

时间限制：1000ms　内存限制：65536kb

通过率：291/441 (65.99%)　正确率：291/960 (30.31%)

题目描述

小伍哥非常喜欢摸鱼，即便是在统计成绩的时候也不例外。机智的小伍哥获取到了同学们在某次练习赛中的得分和罚时，他有若干个问题，每个问题都是以下形式：排名第 xx 名的同学学号是多少。
排名按照得分为第一关键字排序，得分相同的情况下罚时少的同学排名更靠前，罚时也相同的情况下学号小的同学排名更靠前。

输入

第一个数为数据组数 T。
每组数据第一行一个正整数 n，代表学生人数。
接下来 nn 行，每行两个正整数 s,t，代表一名学生的得分和罚时。这 n 名学生的学号依次为 1 n。
接下来一行一个整数 q，代表询问个数。
接下来 q 行每行一个整数 x，代表询问的名次。
T<=10,n<=1000,s<=1000,t<=10000,q<=10,x<=n

输出

对于每个询问，输出一行一个整数，代表这名学生的学号。

输入样例

```
2
2
```

```
100 100
200 200
1
1
2
100 100
100 200
1
2
```

输出样例

```
2
2
```

解题思路

双关键字排序，只需要在把原来用于比较的小于号改成一个自定义的比较函数就可以了。

AC 代码

```
#include<stdio.h>
#include<stdlib.h>
int f(const int*p1,const int *p2){
    if(p1[0]>p2[0])
    return p2[0]-p1[0];
    else if(p1[0]==p2[0]){
        if(p2[1]<p1[1])
        return p1[1]-p2[1];
        else if(p2[1]==p1[1])
        return p2[2]-p1[2];
        else return p1[1]-p2[1];
    }
    else return p2[0]-p1[0];
}
int main(){
    int T,n,s[1010][3],q,x,i,j,k,ans[110],l=0,q1=0;
    scanf("%d",&T);
    for(i=0;i<T;i++){
```

```
        scanf("%d",&n);
        for(j=0;j<n;j++){
            scanf("%d%d",&s[j][0],&s[j][1]);
            s[j][2]=j+1;
        }
        qsort(s,n,sizeof(s[0]),f);
        scanf("%d",&q);
        q1=q1+q;
        for(k=0;k<q;k++){
            scanf("%d",&x);
            printf("%d\n",s[x-1][2]);
        }
    }
}
```

J Tarpe 酋长送快递

时间限制：3000ms　内存限制：65536kb

通过率：499/633 (78.83%)　正确率：499/1707 (29.23%)

题目描述

五一期间，部落的快递员小哥打算放假去陪女票，于是送快递的任务就落在了部落最后的单身狗——Tarpe 酋长身上。小哥在放假之前带走了部落的地图，伤心的酋长觉得自己可能把所有的快递都送错。

现在已知清明节期间有 n 个不同的包裹，需要寄给 n 个不同的地址，酋长想知道每个包裹都送错地址的情况共有多少种?

输入

多组数据，

每组数据一行，一个数 n 表示包裹数和地址数。（1<=n<=21）

输出

对于每行数据，输出一行，表示全部送错的情况数。

输入样例

```
1

3
```

输出样例

```
0

2
```

HINT

```
 错排公式
```

考察知识点

```
递归

难度等级：5
```

解题思路：

这道题目其实是典型的错排问题，利用高中的排列组合（或者度娘）可以很快得到错排公式，

D(n) = (n-1) [D(n-2) + D(n-1)]

特殊地，D(1) = 0, D(2) = 1.

这正是这道题的递归公式。

具体证明过程：https://baike.baidu.com/item/%E9%94%99%E6%8E%92%E5%85%AC%E5%BC%8F/10978508?fr=aladdin

你可能问酋长，要是推不出公式怎么办，那么请写出前 10 个数找找规律，恩对，递归的本质就是找规律。

可能的坑点：

long long 啊 long long

AC 代码：

```
#include<stdio.h>
#include<stdlib.h>
int n = 0;

long long D(int n)
{
    if(n == 1)
    {
        return 0;
    }
    else if( n == 2)
    {
        return 1;
    }
    else
    {
        return (n-1)*(D(n-2) + D(n -1));
    }
}
int main()
{
    //freopen("in3.txt","r",stdin);
    //freopen("out3.txt","w",stdout);
    while(~scanf("%d",&n))
```

```
    {
        printf("%lld\n",D(n));
    }
    //fclose(stdin);
    //fclose(stdout);
}
```