期中考试讲评【1.22】

简介：

先简单讲一下各个题目的思路，并附上姐考试时的代码

1、Digital Reversal

题目描述 展开

输入一个不小于 100 且小于 1000，同时包括小数点后一位的一个浮点数，例如123.4，要求把这个数字翻转过来，变成4.321并输出。

输入格式

无

输出格式

无

输入输出样例

输入 #1 复制	输出 #1 复制
123.4	4.321

这道题是个**字符串处理**的题目，题目虽然说是浮点数类型，其实就是一个无聊的小干扰

你们也都一遍过了，就不多说了

```
#include <iostream>
using namespace std;

int main()
{
    string s;
    cin>>s;
    for(int i=s.length()-1;i>=0;i--) cout<<s[i];
    cout<<endl;
}
```

2、Leap Year Judgment

题目描述 展开

输入一个年份（大于 1582 的整数），判断这一年是否非闰年，如果是输出 1，否则输出 0。

输入格式

无

输出格式

无

输入输出样例

输入 #1 复制	输出 #1 复制
1926	0

模拟题，能被 400 整除或者能被 4 整除但不能被 100 整除即可

（姐比赛的时候还没想清楚这层，导致代码长了一点

```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin>>n;
    if(n%400 == 0) cout<<1<<endl;
    else if(n%100 == 0) cout<<0<<endl;
    else if(n%4 == 0) cout<<1<<endl;
    else cout<<0<<endl;
}
```

3、Triangular Classification

题目描述

展开

给出三条线段 a, b, c 的长度，均是不大于 10000 的整数。打算把这三条线段拼成一个三角形，它可以是什么三角形呢?

- 如果三条线段不能组成一个三角形，输出 `Not triangle` ;
- 如果是直角三角形，输出 `Right triangle` ;
- 如果是锐角三角形，输出 `Acute triangle` ;
- 如果是钝角三角形，输出 `Obtuse triangle` ;
- 如果是等腰三角形，输出 `Isosceles triangle` ;
- 如果是等边三角形，输出 `Equilateral triangle` 。

如果这个三角形符合以上多个条件，请分别输出，并用换行符隔开。

输入格式

无

输出格式

无

输入输出样例

无

模拟题，根据三条边的长度关系判断是哪种三角形即可

记得同时满足多个条件要按顺序输出

一个小坑：如果判断不构成三角形的话应该马上结束，不然可能会被判断成钝角三角形或者等腰三角形

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main()
{
    int a[3];
    cin>>a[0]>>a[1]>>a[2];
    sort(a,a+3);
    if(a[0]+a[1] <= a[2]){
        cout<<"Not triangle"<<endl;
        return 0;
    }
    if(a[0]*a[0]+a[1]*a[1]>a[2]*a[2]) cout<<"Acute triangle"<<endl;
    if(a[0]*a[0]+a[1]*a[1]==a[2]*a[2]) cout<<"Right triangle"<<endl;
    if(a[0]*a[0]+a[1]*a[1]<a[2]*a[2]) cout<<"Obtuse triangle"<<endl;
    if(a[0] == a[1] or a[1] == a[2] or a[0] == a[2]) cout<<"Isosceles triangle"<<endl;
    if(a[0] == a[1] and a[1] == a[2] and a[0] == a[2]) cout<<"Equilateral triangle"<<endl;
    return 0;
}
```

4、Prime Pocket

题目描述

展开

小A 有一个质数口袋，里面可以装各个质数。他从 2 开始，依次判断各个自然数是不是质数，如果是质数就会把这个数字装入口袋。口袋的负载量就是口袋里的所有数字之和。但是口袋的承重量有限，不能装得下总和超过 $L(1 \leq L \leq 100000)$ 的质数。给出 L，请问口袋里能装下几个质数？将这些质数从小往大输出，然后输出最多能装下的质数个数，所有数字之间有一空行。

输入格式

无

输出格式

无

输入输出样例

输入 #1 复制

```
100
```

输出 #1 复制

```
2
3
5
7
11
13
17
19
23
9
```

模拟题，从 2 开始递增遍历，判断如果是素数，就尝试放进口袋，如果口袋满了，就输出个数；没满，就输出该素数

这里可以学一下埃氏筛，直接得出 0-n 哪些数是素数，复杂度为 O(nloglogn)

（当然还有 O(n)的方式，叫欧拉筛，代码麻烦一点，学埃氏筛也差不多啦

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main()
{
    int a[100002],i,j;
    //埃氏筛，判断0-100000哪些是素数
    for(i=0;i<100002;i++) a[i] = 1;
    a[0] = a[1] = 0;
    for(i=2;i<100002;i++){
        if(a[i] == 1){
            for(j=2;i*j<100002;j++){
                a[i*j] = 0;
            }
        }
    }
    //正式开始
    int t,sum = 0,count = 0;
    cin>>t;
    for(i=2;;i++){
        if(a[i] == 1){
            sum += i;
            if(sum > t) break;
            cout<<i<<endl;
            count++;
        }
    }
    cout<<count<<endl;
    return 0;
}
```

5、Craft Production

题目描述

展开

现有一个长宽高分别为 $w,x,h(1 \le w,x,h \le 20)$ 组成的实心玻璃立方体，可以认为是由 $1 \times 1 \times 1$ 的数个小方块组成的，每个小方块都有一个坐标 (i,j,k)。现在需要进行 $q(q \le 100)$ 次切割。每次切割给出 $(x_1,y_1,z_1),(x_2,y_2,z_2)$ 这 6 个参数，保证 $x_1 \le x_2$，$y_1 \le y_2$，$z_1 \le z_2$；每次切割时，使用激光工具切出一个立方体空洞，空洞的壁平行于立方体的面，空洞的对角点就是给出的切割参数的两个点。

换句话说，所有满足 $x_1 \le i \le x_2$，$y_1 \le j \le y_2$，$z_1 \le k \le z_2$ 的小方块 (i,j,k) 的点都会被激光蒸发。例如有一个 4×4×4 的大方块，其体积为 64；给出参数 (1,1,1),(2,2,2) 时，中间的 8 块小方块就会被蒸发，剩下 56 个小方块。现在想知道经过所有切割操作后，剩下的工艺品还剩下多少格小方块的体积?

输入格式

第一行四个整数 w,x,h,q。

接下来 q 行，每行六个整数 $(x_1,y_1,z_1),(x_2,y_2,z_2)$

输出格式

输出一个整数表示答案。

输入输出样例

输入 #1 复制

```
4 4 4 1
1 1 1 2 2 2
```

输出 #1 复制

```
56
```

模拟题，三重循环对三维数组进行维护即可

（维护：修改某个位置的数据

值得注意的是，三维数组开大一格，比如说上面输入 4 4 4，数组开成 a[5][5][5]，然后舍弃下标为 0 的，会方便很多

（当然你不这么做也是可以的

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main()
{
    int i,j,k,a,b,c,t,o,p,q,r,s,u;
    cin>>a>>b>>c>>t;
    int m[a][b][c];
    for(i=0;i<a;i++)
        for(j=0;j<b;j++)
            for(k=0;k<c;k++)
                m[i][j][k] = 1;
    while(t--){
        cin>>o>>p>>q>>r>>s>>u;
        for(i=o-1;i<r;i++)
            for(j=p-1;j<s;j++)
                for(k=q-1;k<u;k++)
                    m[i][j][k] = 0;
    }
    int sum = 0;
    for(i=0;i<a;i++)
        for(j=0;j<b;j++)
            for(k=0;k<c;k++)
                if(m[i][j][k] == 1)
                    sum++;
    cout<<sum<<endl;
}
```

6、Distance Function

题目描述 展开

给出平面坐标上不在一条直线上三个点坐标 $(x_1,y_1),(x_2,y_2),(x_3,y_3)$，坐标值是实数，且的绝对值不超过 100.00，求围成的三角形周长。保留两位小数。

输入格式

无

输出格式

无

输入输出样例

输入 #1 复制	输出 #1 复制
0 0 0 3 4 0	12.00

模拟题，使用 cmath 库的 sqrt 开根号函数即可求出三边，然后相加即可，记得输出保留两位小数

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <iomanip>
using namespace std;

int main()
{
    double s1,s2,s3,s4,s5,s6;
    cin>>s1>>s2>>s3>>s4>>s5>>s6;
    double x = sqrt((s1-s3)*(s1-s3)+(s2-s4)*(s2-s4));
    double y = sqrt((s3-s5)*(s3-s5)+(s6-s4)*(s6-s4));
    double z = sqrt((s1-s5)*(s1-s5)+(s2-s6)*(s2-s6));
    double sum = x+y+z;
    cout<<setiosflags(ios::fixed)<<setprecision(2)<<sum<<endl;
}
```

7、Well Matched Adversary

题目描述

展开

现有 $N(N \leq 1000)$ 名同学参加了期末考试，并且获得了每名同学的信息：姓名（不超过 8 个字符的字符串，没有空格）、语文、数学、英语成绩（均为不超过 150 的自然数）。如果某对学生 <i,j> 的每一科成绩的分差都不大于 5，且总分分差不大于 10，那么这对学生就是“旗鼓相当的对手”。现在我们想知道这些同学中，哪些是“旗鼓相当的对手”？请输出他们的姓名。

所有人的姓名是按照字典序给出的，输出时也应该按照字典序输出所有对手组合。也就是说，这对组合的第一个名字的字典序应该小于第二个；如果两个组合中第一个名字不一样，则第一个名字字典序小的先输出；如果两个组合的第一个名字一样但第二个名字不同，则第二个名字字典序小的先输出。

输入格式

无

输出格式

无

输入输出样例

输入 #1 复制

```
3
fafa 90 90 90
lxl 95 85 90
senpai 100 80 91
```

输出 #1 复制

```
fafa lxl
lxl senpai
```

模拟题，结构体（或多维数组）保存数据，然后遍历对比，满足单科分差不大于 5，总分分差不大于 10，就输出双方名字即可

abs 函数：传入一个参数，返回参数的绝对值

```cpp
#include <iostream>
#include <cmath>
#include <algorithm>
#include <iomanip>
using namespace std;

struct node{
    string name;
    int yuwen,shuxue,yinyu;
};

int main()
{
    int i,j,n;
    cin>>n;
    node a[n];
    for(i=0;i<n;i++){
        cin>>a[i].name>>a[i].yuwen>>a[i].shuxue>>a[i].yinyu;
    }
    for(i=0;i<n;i++){
        for(j=i+1;j<n;j++){
            if(abs(a[i].yuwen-a[j].yuwen)<6
            and abs(a[i].shuxue-a[j].shuxue)<6
            and abs(a[i].yinyu-a[j].yinyu)<6
            and abs(a[i].yuwen+a[i].shuxue+a[i].yinyu-a[j].yuwen-a[j].yinyu-a[j].shuxue)<11)
                cout<<a[i].name<<" "<<a[j].name<<endl;
        }
    }
}
```

8、Find The Kth Smallest Number

题目描述 展开

输入 $n(n < 5000000$ 且 n 为奇数) 个数字 $a_i(0 < a_i < 10^9)$，输出这些数字的第 k 小的数。最小的数是第 0 小。

输入格式

无

输出格式

无

输入输出样例

输入 #1 复制

```
5 1
4 3 2 1 5
```

输出 #1 复制

```
2
```

排序题：60 分水题，100 分较难

60 分做法：快排，然后输出第 k 位（时间复杂度为 O(nlogn)

100 分做法：如果了解快排的思想（二分）的话，快排一步会将原数组分成三部分，其中左边部分的任意数字均小于等于中间数字，右边部分的任意数字均大于等于中间数字，因此可以发现：如果 k>中间数字的位置，则第 k 小的数在右边部分；如果 k<中间数字的位置，则在左边部分；如果 k==中间数字的位置，则输出该数字，结束。因此只需对相应部分继续快排，重复上述过程即可。时间复杂度为 O(n)

100 分更简单做法：用 STL 里一个叫做 nth_element 的东西：

nth_element(a,a+k,a+n)：将数组 a 中第 k 小的数放在 a[k]中，但是不保证其它位置有序，然后输出 a[k]即可，时间复杂度为 O(n)

提示，scanf 比 cin 快，在 TLE 的题目中用 scanf 可节约时间

```cpp
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int n,k,i;
    scanf("%d%d",&n,&k);
    int a[n];
    for(i=0;i<n;i++) scanf("%d",&a[i]);
    nth_element(a,a+k,a+n);
    printf("%d\n",a[k]);
}
```

9、Who Was First Runner-Up

题目描述 展开

有 $2^n(n \leq 7)$ 个国家参加世界杯决赛圈且进入淘汰赛环节。我经知道各个国家的能力值，且都不相等。能力值高的国家和能力值低的国家踢比赛时高者获胜。1 号国家和 2 号国家踢一场比赛，胜者晋级。3 号国家和 4 号国家也踢一场，胜者晋级……晋级后的国家用相同的方法继续完成赛程，直到决出冠军。给出各个国家的能力值，请问亚军是哪个国家？

输入格式

无

输出格式

无

输入输出样例

输入 #1 复制	输出 #1 复制
3 4 2 3 1 10 5 9 7	1

模拟题，正赛的最后一题，操作比较啰嗦，但是题意挺明显的

先将所有国家的能力值保存进 vector 中（用结构体同时保存国家号），之后每两个国家进行比较，胜者放入一个新 vector 中，比较完这个 vector 后，看新的 vector，如果长度为 2，则已经是决赛，输出能力值低的国家的国家号即可；如果长度不为 2，则继续比较生成新的 vector，直到新 vector 长度为 2 为止。

（不得不说这是个不太好的思路，但是考试嘛时间比较赶

一个更好的思路：用队列，每次头部出队两个元素，把强者入队，当队列长度为 2 时，则已经是决赛，出队两个元素，输出能力值低的国家的国家号即可。

```cpp
#include <iostream>
#include <cmath>
#include <algorithm>
#include <vector>
using namespace std;

struct node{
    int num;
    int p;
};

int main()
{
    int i,n,k;
    cin>>n;
    int num = (int)pow(2,n);
    vector<node> v,u;
    for(i=0;i<num;i++){
        cin>>k;
        node no;
        no.num = i+1;
        no.p = k;
        v.push_back(no);
    }
    if(n == 1) goto tt;
    th:
    for(i=0;i<v.size();i+=2){
        if(v[i].p>v[i+1].p) u.push_back(v[i]);
        else u.push_back(v[i+1]);
    }
    v = u;
    u.clear();
    if(v.size()>2) goto th;
    tt:
    if(v[0].p>v[1].p) cout<<v[1].num;
    else cout<<v[0].num;
    cout<<endl;
}
```

10、Ringed Genesis (附加题)

题目背景

展开

Enzyme runs through the Ringed Genesis，just like Rabbit runs through a Ring.

题目描述

有一个长长的环，环由 n 个格子首尾相接形成，依次编号 0 至 $n-1$。

还有一种动物——兔子。兔子的步长为 k。若兔子当前在第 i 个格子，那么下一秒它将跳到第 $(i+k) \bmod n$ 个格子。

现在有 m 只兔子，第 i 只兔子的初始格子为第 p_i 个格子。随着时间的流逝，有些格子被兔子经过了，有些却一直没有被兔子经过。

你需要求出的是，有多少个格子永远不可能被兔子经过。

输入格式

从标准输入中读取数据。

第一行，三个正整数 n, m, k，表示环长，兔子数，步长。

第二行，m 个非负整数 $p_1, p_2, \ldots, p_m$，表示兔子的初始格子。

输出格式

输出数据至标准输出中。

共一行，一个整数，表示答案。

输入输出样例

输入 #1 复制

```
4 2 2
0 1
```

输出 #1 复制

```
0
```

输入 #2 复制

```
4 2 2
0 2
```

输出 #2 复制

```
2
```

说明/提示

子任务 1（10%）：$k=1$。

子任务 2（20%）：$k|n$，也即 $\gcd(k,n)=k$。

子任务 3（25%）：$1 \le n \le 1000$，$1 \le m \le 1000$。

子任务 4（45%）：无特殊限制。

对于全部数据，$1 \le n \le 10^6$，$1 \le m \le 10^6$，$1 \le k \le n$。

模拟题，另一场比赛的一道普及-难度的题

主要是找来凑够 10 道题好看

这道题还没人做，那就你们自己思考一下趴