**宋老师**

**上课要求：**

1、上课认真听讲；

**2、课下作业按时完成；**

3、不懂的要问，**懂了的要记住**；

（1）**样例代码要能做到不看笔记写出程序**；

（2）作业遇到不会的，应当及时去复习电子笔记；

4、出现遗忘及时复习；

5、锻炼自己的**调错能力**；

提升：代码掌握的熟练度，**熟能生巧**。

**提升：首次提交代码的正确率！☆☆☆☆☆**

**做完题：**

1. 静态阅读代码；

2. 学会**出**样例数据、反复测试、思考题目是否有特殊情况！

看数据量、数据范围，计算时间复杂度！

**常规作业：**

**1.  在题库提交过的，有错的题目，又订正出来了，一定要有总结。☆☆☆☆☆**

**2. 每个月的月赛请及时参加，目前参加A组。**

**3. 做对的题目，如果有题解，一定要看题解。☆☆☆☆☆**

**通过参考解法，优化自己的代码，提升编程水平。**

**做竞赛题一定要知道的几个数据：**

**1. 1秒最多循环108次。**

**2. 在函数中定义数组，数组最大开105。**

**在main的外面定义数组，512MB最大开108，一般控制在106，或者107范围内。**

**3. int的最大数是231-1，是一个十位数。**

**long long的最大数是263-1，是一个19位的数。**

**比赛如果部分数据点时间超限，还是有一部分分数的。**

**但如果内存超限（数组开太大）一定0分。**

等差数列求和公式：n \* (a1 + an) / 2，n代表总项数，a1：第1项的值，an：第n项的值。

一、基础巩固复习

A. 七位完全平方数

#include <bits/stdc++.h>

using namespace std;

/\*

1. 枚举哪些数的平方是7位数

i => sqrt(1000000)~sqrt(9999999)

i \* i，一定是平方数，也一定是7位数

2. 判断i \* i是否各个位不同

短除法，用数组标记的方法来判断

\*/

int c[20];//计数数组，记录每个位上的数出现的次数

int cnt;//统计当前满足条件的数出现的数量

int main() {

    int x;

    bool f;

    for(int i = sqrt(1000000); i <= sqrt(9999999); i++) {

        x = i \* i;

        //判断x各个位不同

        memset(c, 0, sizeof(c));

        f = false;//假设各个位都是不相同的，不存在任意的两位是一样的

        //短除法拆位

        while(x != 0) {

            c[x % 10]++;

            if(c[x % 10] > 1) f = true;//发现有两位相同的情况

            x /= 10;

        }

        //如果各个位确实都是不相同

        if(f == false) {

            cout << i \* i;

            cnt++;

            if(cnt % 10 == 0) cout << endl;

            else cout << " ";

        }

    }

    return 0;

}

B. 统计每个数出现的次数

#include <bits/stdc++.h>

using namespace std;

/\*

10

2 8 1 2 3 3 6 1 1 1000

1. 排序：相同的数会排在一起

1 1 1 2 2 3 3 6 8 1000

2. 数一数：连续相等的数出现的次数

1 3

2 2

3 2

6 1

8 1

1000 1

\*/

int a[1010];

int n, c = 0;

int main() {

    cin >> n;

    for(int i = 1; i <= n; i++) {

        cin >> a[i];

    }

    //排序

    sort(a + 1, a + n + 1);

    //统计

    for(int i = 1; i <= n; i++) {

        c++;

        //到了最后一个||下一个不符合连续相等的条件

        if(i == n || a[i] != a[i + 1]) {

            //说明当前这个数，是连续相等数的最后一个

            cout << a[i] << " " << c << endl;

            c = 0;//计数器清零

        }

    }

    return 0;

}

连续性统计问题：统计连续相等的数、连续递增的数、连续的偶数、连续非素数出现的次数

**连续性统计代码模版：**

for(int i => 枚举每个数){

    if(a[i]符合连续性的特点){//连续相等、递增不需要判断，连续偶数、非素数就要判断

        c++;

        if(到了最后一个 || 下一个不符合连续性的特点){

            输出结果;

            c = 0;

        }

    }

}

D. 黑色星期五（friday）

#include <bits/stdc++.h>

using namespace std;

/\*

求：1900年1月1日~1900+n-1年12月31日，

13号出现在周一~周日的次数。

方法：按天枚举。

\*/

int m[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int n;

int r[20];//统计13号在周一~周日的次数

int w = 1;//星期几

int main() {

    cin >> n;

    //枚举年

    for(int i = 1900; i <= 1900 + n - 1; i++) {

        //判闰年

        if(i % 4 == 0 && i % 100 != 0 || i % 400 == 0) m[2] = 29;

        else m[2] = 28;

        //枚举月

        for(int j = 1; j <= 12; j++) {

            //枚举日

            for(int k = 1; k <= m[j]; k++) {

                //统计13号

                if(k == 13) {

                    r[w]++;

                }

                w++;//星期

                if(w == 8) w = 1;

            }

        }

    }

    //输出

    cout << r[6] << " " << r[7] << " ";

    for(int i = 1; i <= 5; i++) {

        cout << r[i] << " ";

    }

    return 0;

}

declare：定义

a was not declared in this scope：变量a在这个范围内没有定义过

F. 简单计算器

#include <bits/stdc++.h>

using namespace std;

/\*

给定一个表达式，计算表达式的结果，注意结果在long long的范围内。

表单内的每个值都在1000的范围内。

30+10\*20\*30-40

1. 分解出运算数和运算符，并存储

2. 扫描运算符数组

A.遇到\*，将这一位运算数和下一位相乘，结果存下一位，本位清零

B.遇到-，将下一位运算数，变成相反数。

3. 求和。

\*/

string s;

long long a[20], c1; //运算数

char f[20], c2; //运算符

long long ans = 0;//运算结果

int t = 0;//分解出来的每个运算数

//判断一个字符是否是数字字符

bool isnum(char c) {

    if(c >= '0' && c <= '9') return true;

    else return false;

}

int main() {

    cin >> s;

    //将连续的数字分解为一个运算数

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

        //如果是数字字符

        if(isnum(s[i])) {

            t = t \* 10 + (s[i] - '0');//将连续的整数拼接出来

            //判断连续的数字字符结束了

            if(i == s.size() - 1 || !isnum(s[i + 1])) {

                c1++;//运算数多了一个

                a[c1] = t;//将运算数存起来

                t = 0;

            }

        } else {

            c2++;//运算符多了一个，并存起来

            f[c2] = s[i];

        }

    }

    //计算：枚举运算符数组

    for(int i = 1;i <= c2;i++){

        if(f[i] == '\*'){

            a[i + 1] = a[i] \* a[i + 1];

            a[i] = 0;

        }else if(f[i] == '-'){

            a[i + 1] = -a[i + 1];

        }

    }

    //将运算数数组中的所有数求和

    for(int i = 1;i <= c1;i++){

        ans += a[i];

    }

    cout<<ans;

    return 0;

}