长 沙 学 院

课程设计说明书

|  |  |
| --- | --- |
| **题目** | **程序设计算法综合训练** |
| **院(部)** | **计算机工程与应用数学学院** |
| **专业(班级)** | **20软件二班** |
| **姓名** | **李佳骏** |
| **学号** | **B20190103224** |
| **指导教师** | **黄娟** |
| **起止日期** | **2021.12.27-2021.12.31** |

课程设计任务书

**课程名称：**程序设计基础课程设计

**设计题目：**程序设计算法综合训练

**已知技术参数和设计要求：**

1. **基础练习（共6题）**

**1、【BEGIN-04】[Fibonacci数列](http://lx.lanqiao.cn/problem.page?gpid=T4)**

**<http://lx.lanqiao.cn/problem.page?gpid=T4>**

Fibonacci数列的递推公式为：Fn=Fn-1+Fn-2，其中F1=F2=1。当n比较大时，Fn也非常大，现在我们想知道，Fn除以10007的余数是多少。

**2、【BASIC-1】闰年判断**

**<http://lx.lanqiao.cn/problem.page?gpid=T5>**

给定一个年份，判断这一年是不是闰年。当以下情况之一满足时，这一年是闰年：1. 年份是4的倍数而不是100的倍数；2. 年份是400的倍数。3. 其他的年份都不是闰年。

**3、【BASIC-4】数列特征**

**<http://lx.lanqiao.cn/problem.page?gpid=T8>**

给出n个数，找出这n个数的最大值，最小值，和。

**4、【BASIC-5】查找整数**

**<http://lx.lanqiao.cn/problem.page?gpid=T9>**

给出一个包含n个整数的数列，问整数a在数列中的第一次出现是第几个。

**5、【BASIC-6】杨辉三角形**

**<http://lx.lanqiao.cn/problem.page?gpid=T10>**

杨辉三角形又称Pascal三角形，它的第i+1行是(a+b)i的展开式的系数。它的一个重要性质是：三角形中的每个数字等于它两肩上的数字相加。下面给出了杨辉三角形的前4行：

1

1 1

1 2 1

1 3 3 1

给出n，输出它的前n行。

**6、【BASIC-13】数列排序**

**<http://lx.lanqiao.cn/problem.page?gpid=T52>**

给定一个长度为n的数列，将这个数列按从小到大的顺序排列。1<=n<=200

**二、算法训练（共12题）**

**7、【ALGO-955】 P0701**

**<http://lx.lanqiao.cn/problem.page?gpid=T2652>**

编写一个函数RegularPlural，其功能是实现一个英文单词的复数形式。复数的规则为：

（1） 如果单词末尾为s,x,z,ch或sh，则在后面加es

（2） 如果单词末尾为y，且前一个字母为辅音（除a, e, i, o, u以外的其它情况），则把y改成ies。

（3） 如果是其它情形，一律在后面加s。

编写测试程序，输入一个长度小于20的单词，输出该单词的复数形式。

**8、【ALGO-956】 P0702**

**<http://lx.lanqiao.cn/problem.page?gpid=T2653>**

在C语言中，有一个strcmp函数，其功能是比较两个字符串s1和s2。请编写一个你自己的字符串比较函数my\_strcmp，来实现strcmp函数的类似功能。如果s1=s2，则返回0；否则返回s1 与s2 第一个不同字符的差值（如果s1<s2，该差值是一个负数；如果s1>s2，该差值是一个正数）。编写测试程序，输入两个长度小于1000的字符串（可能包含有空格，且长度不一定相等），然后调用my\_strcmp函数来进行比较，并输出返回结果。

**9、【ALGO-957】 P0703**

**<http://lx.lanqiao.cn/problem.page?gpid=T2654>**

一个整数的反置数指的是把该整数的每一位数字的顺序颠倒过来所得到的另一个整数。如果一个整数的末尾是以0结尾，那么在它的反置数当中，这些0就被省略掉了。比如说，1245的反置数5421，而1200的反置数是21。请编写一个程序，输入两个整数，然后计算这两个整数的反置数之和sum，然后再把sum的反置数打印出来。例如，如果用户输入：435和754，则输出结果为199。要求：由于在本题中需要多次去计算一个整数的反置数，因此必须把这部分代码抽象为一个函数的形式。

**10、【ALGO-958】 P0704**

**<http://lx.lanqiao.cn/problem.page?gpid=T2655>**

一个数如果从左往右读和从右往左读数字是完全相同的，则称这个数为回文数，比如898,1221,15651都是回文数。编写一个程序，输入两个整数min和max，然后对于min~max之间的每一个整数（包括min和max），如果它既是一个回文数又是一个质数，那么就把它打印出来。要求，回文数和质数的判断都必要要用函数的形式来实现。

**11、【ALGO-941】 P0601**

**<http://lx.lanqiao.cn/problem.page?gpid=T2626>**

编写一个程序，先输入一个字符串str（长度不超过20），再输入单独的一个字符ch，然后程序会把字符串str当中出现的所有的ch字符都删掉，从而得到一个新的字符串str2，然后把这个字符串打印出来。

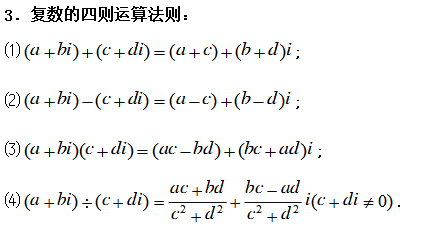
**12、【ALGO-142】P1103 复数运算**

**<http://lx.lanqiao.cn/problem.page?gpid=T372>**

编程实现两个复数的运算。

要求（1）定义一个结构体类型来描述复数。

　　（2）复数之间的加法、减法、乘法和除法分别用不用的函数来实现。



**13、【ALGO-79】 删除数组零元素**

**<http://lx.lanqiao.cn/problem.page?gpid=T201>**

从键盘读入n个整数放入数组中，编写函数CompactIntegers，删除数组中所有值为0的元素，其后元素向数组首端移动。注意，CompactIntegers函数需要接受数组及其元素个数作为参数，函数返回值应为删除操作执行后数组的新元素个数。输出删除后数组中元素的个数并依次输出数组元素。

**14、【ALGO-467】 大整数加法**

**<http://lx.lanqiao.cn/problem.page?gpid=T1424>**

任意输入两个正整数a,b，求两数之和。（注：本题会输入超过32位整数限制的大整数）

建议使用字符数组实现。

**15、【ALGO-53】 最小乘积(基本型)**

**<http://lx.lanqiao.cn/problem.page?gpid=T133>**

给两组数，各n个。

请调整每组数的排列顺序，使得两组数据相同下标元素对应相乘，然后相加的和最小。要求程序输出这个最小值。

例如两组数分别为:1 3 -5和-2 4 1

　　那么对应乘积取和的最小值应为：

(-5) \* 4 + 3 \* (-2) + 1 \* 1 = -25

**16、【ALGO-561】 矩阵运算**

**<http://lx.lanqiao.cn/problem.page?gpid=T1729>**

给定一个n\*n的矩阵A，求A+AT的值。其中AT表示A的转置。

**17、【ALGO-551】 百鸡百钱**

**<http://lx.lanqiao.cn/problem.page?gpid=T1675>**

我国古代数学家张丘建在《算经》一书中提出的数学问题：鸡翁一值钱五，鸡母一值钱三，鸡雏三值钱一。百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？

**18、【ALGO-538】 数据传递加密**

**<http://lx.lanqiao.cn/problem.page?gpid=T1635>**

某个公司传递数据，数据是四位整数，在传递过程中需要进行加密的，加密规则如下：每位数字都加上5,然后除以10的余数代替该位数字。再将新生成数据的第一位和第四位交换，第二位和第三位交换。要求输入4位整数，输出加密后的4位整数。比如：输入一个四位整数1234，则输出加密结果为9876。

**三、算法提高（共4题）**

**19、【ADV-1082】 心形**

**<http://lx.lanqiao.cn/problem.page?gpid=T2529>**

根据给出的最大宽度，输出心形（注意空格，行末不加多余空格）。

**20、【ADV-658】 字符串查找**

**<http://lx.lanqiao.cn/problem.page?gpid=T1728>**

给定两个字符串a和b，查找b在a中第一次出现的位置。

如a=“hello world”，b=“world”时，b第一次出现是在a的第7个字符到第11个字符，按照C++的习惯，位置从0开始编号，所以b在a中第一次出现的位置为6。

**21、【ADV-789】 校门外的树**

**<http://lx.lanqiao.cn/problem.page?gpid=T1930>**

某校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置；数 轴上的每个整数点，即0，1，2，……，L，都种有一棵树。

由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树 都移走后，马路上还有多少棵树。

**22、【ADV-100】 第二大整数**

**<http://lx.lanqiao.cn/problem.page?gpid=T211>**

编写一个程序，读入一组整数（不超过20个），当用户输入0时，表示输入结束。然后程序将从这组整数中，把第二大的那个整数找出来，并把它打印出来。说明：（1）0表示输入结束，它本身并不计入这组整数中。（2）在这组整数中，既有正数，也可能有负数。（3）这组整数的个数不少于2个。

**四、历届试题（共3题）**

**23、【PREV-151】 四平方和**

**<http://lx.lanqiao.cn/problem.page?gpid=T2780>**

四平方和定理，又称为拉格朗日定理：每个正整数都可以表示为至多4个正整数的平方和。　　如果把0包括进去，就正好可以表示为4个数的平方和。

　　比如：

　　5 = 0^2 + 0^2 + 1^2 + 2^2

　　7 = 1^2 + 1^2 + 1^2 + 2^2

　　（^符号表示乘方的意思）

　　对于一个给定的正整数，可能存在多种平方和的表示法。

　　要求你对4个数排序：

　　0 <= a <= b <= c <= d

　　并对所有的可能表示法按 a,b,c,d 为联合主键升序排列，最后输出第一个表示法

**24、【PREV-286】 错误票据**

**<http://lx.lanqiao.cn/problem.page?gpid=T2916>**

某涉密单位下发了某种票据，并要在年终全部收回。每张票据有唯一的ID号。全年所有票据的ID号是连续的，但ID的开始数码是随机选定的。因为工作人员疏忽，在录入ID号的时候发生了一处错误，造成了某个ID断号，另外一个ID重号。你的任务是通过编程，找出断号的ID和重号的ID。假设断号不可能发生在最大和最小号。

**25、【PREV-343】 密码发生器**

**<http://lx.lanqiao.cn/problem.page?gpid=T3042>**

在对银行账户等重要权限设置密码的时候，我们常常遇到这样的烦恼：如果为了好记用生日吧，容易被破解，不安全；如果设置不好记的密码，又担心自己也会忘记；如果写在纸上，担心纸张被别人发现或弄丢了。

这个程序的任务就是把一串拼音字母转换为6位数字（密码）。我们可以使用任何好记的拼音串(比如名字，王喜明，就写：wangximing)作为输入，程序输出6位数字。

　　变换的过程如下：

第一步. 把字符串6个一组折叠起来，比如wangximing则变为：

　　 wangxi

　 ming

　　第二步. 把所有垂直在同一个位置的字符的ascii码值相加，得出6个数字，如上面的例子， 则得出：

　　228 202 220 206 120 105

　　第三步. 再把每个数字“缩位”处理：就是把每个位的数字相加，得出的数字如果不是一位数字， 就再缩位，直到变成一位数字为止。例如: 228 => 2+2+8=12 => 1+2=3

　　上面的数字缩位后变为：344836, 这就是程序最终的输出结果！

**各阶段具体要求：**

1、完成基础练习

熟悉练习系统，能够完成入门训练的题目，并正确提交，并完成相应的文档总结。

2、完成算法训练

能够进行简单的算法分析，完成入基础练习的题目，并正确提交，并完成相应的文档总结。

3、完成算法提高

能够进行一定算法分析设计，完成入算法训练的题目，并正确提交，并完成相应的文档总结。

4、完成历届试题

能够综合运用所学知识，进行算法分析与设计，完成入历届试题的题目，并正确提交，并完成相应的文档总结。

**设计工作量：**

（1）编码测试：在系统中正确提交代码并得分。

（2）设计报告：**要求撰写完成习题的题解分析文档，要求①说明算法思路，②绘制流程图（至少5个），③源代码，④测试结果截图，⑤系统提交得分截图**

**工作计划：**

解题进度安排如下，文档编写同步进行；

1. 基础练习（2课时）

2. 算法训练（8课时）

3. 算法提高（4课时）

4. 历届试题（2课时）

5. 答辩考核（4课时）

**注意事项**

* + **提交文档**
* 长沙学院课程设计任务书 （每学生1份）
* 长沙学院课程设计论文 （每学生1份）
* 长沙学院课程设计鉴定表 （每学生1份）

目录

[一. 基础练习 10](#_Toc30366)

[1、【BEGIN-04】Fibonacci数列 10](#_Toc23506)

[2、【BASIC-1】闰年判断 12](#_Toc12397)

[3、【BASIC-4】数列特征 15](#_Toc167)

[4、【BASIC-5】查找整数 17](#_Toc21960)

[5、【BASIC-6】杨辉三角形 19](#_Toc13609)

[6、【BASIC-13】数列排序 21](#_Toc10791)

[二. 算法训练 22](#_Toc17276)

[7、【ALGO-955】P0701 22](#_Toc14721)

[8、【ALGO-956】P0702 24](#_Toc20625)

[9、【ALGO-957】P0703 25](#_Toc8823)

[10、【ALGO-958】P0704 26](#_Toc8242)

[11、【ALGO-941】P0601 28](#_Toc25260)

[12、【ALGO-142】P1103 29](#_Toc26223)

[13、【ALGO-79】删除数组零元素 31](#_Toc16127)

[14、【ALGO-467】大整数加法 32](#_Toc31095)

[15、【ALGO-53】最小乘积(基本型) 34](#_Toc28294)

[16、【ALGO-561】矩阵运算 35](#_Toc3869)

[17、【ALGO-551】百鸡百钱 36](#_Toc9856)

[18、【ALGO-538】数据传递加密 37](#_Toc19349)

[三. 算法提高 38](#_Toc22154)

[19、【ADV-1082】心形 38](#_Toc21308)

[20、【ADV-658】字符串查找 39](#_Toc8165)

[21、【ADV-789】校门外的树 40](#_Toc24413)

[22、【ADV-100】第二大整数 42](#_Toc10130)

[四. 历届试题 43](#_Toc5708)

[23、【PREV-151】四平方和 43](#_Toc8835)

[24、【PREV-286】错误票据 45](#_Toc10589)

[25、【PREV-343】密码发生器 46](#_Toc13374)

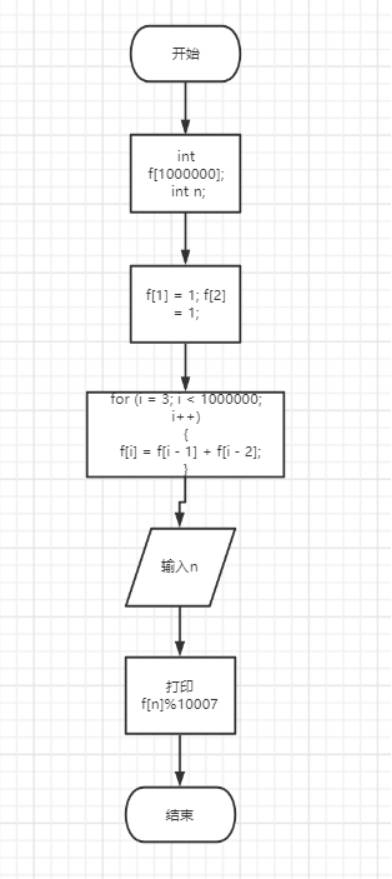
1. 基础练习

**1、【BEGIN-04】[Fibonacci数列](http://lx.lanqiao.cn/problem.page?gpid=T4)**

Fibonacci数列的递推公式为：Fn=Fn-1+Fn-2，其中F1=F2=1。当n比较大时，Fn也非常大，现在我们想知道，Fn除以10007的余数是多少。

**算法思路:** 在本题中，答案是要求Fn除以10007的余数，因此我们只要能算出这个余数即可，而不需要先计算出Fn的准确值，再将计算的结果除以10007取余数，直接计算余数往往比先算出原数再取余简单。

**流程图:**



**源代码:**

#include "stdio.h"

#include "stdlib.h"

int main(){

int i, n;

scanf("%d", &n);

int \*f = (int \*)malloc(sizeof(int) \* (n + 1));

f[0] = f[1] = 1;

for (i = 2; i < n; i++) {

f[i] = (f[i-1] + f[i-2]) % 10007;

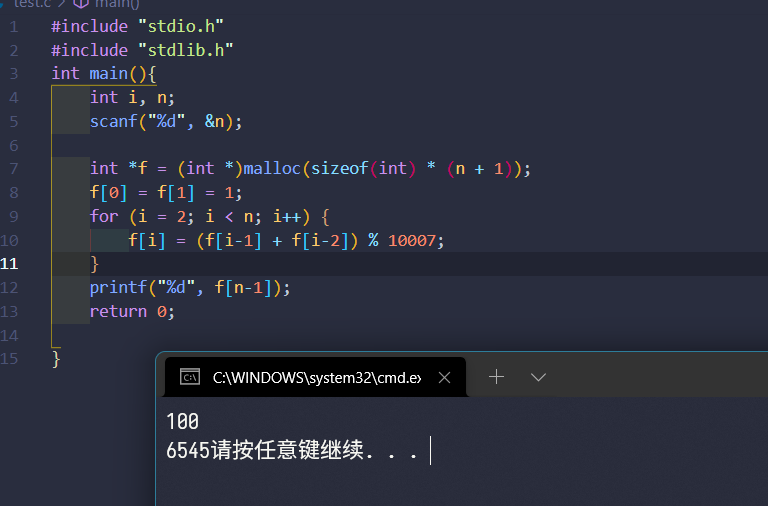
}

printf("%d", f[n-1]);

return 0;

}

**运行截图:**

****

**提交记录:**

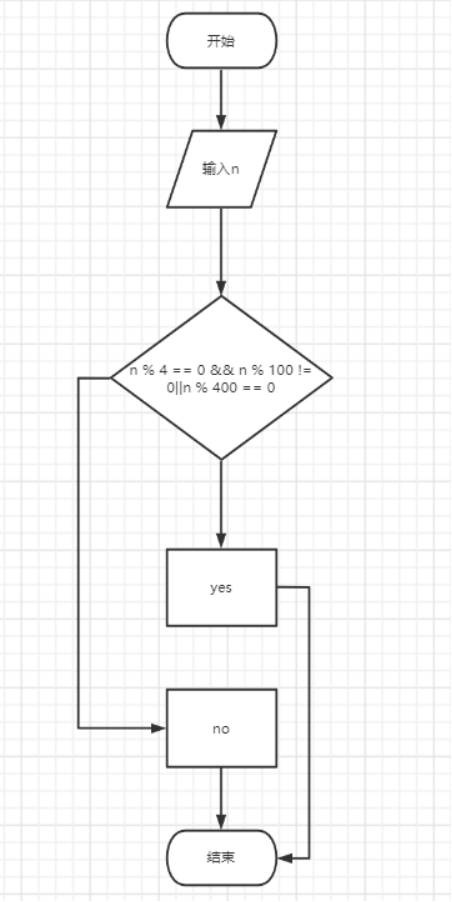
****

**2、【BASIC-1】闰年判断**

给定一个年份，判断这一年是不是闰年。当以下情况之一满足时，这一年是闰年：1. 年份是4的倍数而不是100的倍数；2. 年份是400的倍数。3. 其他的年份都不是闰年。

**算法思路:根据闰年的定义写出判断条件即可**

**流程图:**



**源代码:**

#include <stdio.h>

int main()

{

int year;

scanf("%d", &year);

if (year % 400 == 0 || year % 4 == 0 && year % 100 != 0)

printf ("yes");

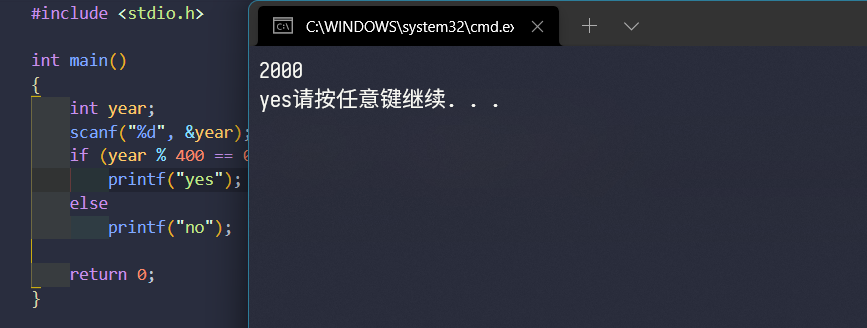
else

printf ("no");

return 0;

}

**运行截图:**



**提交记录:**

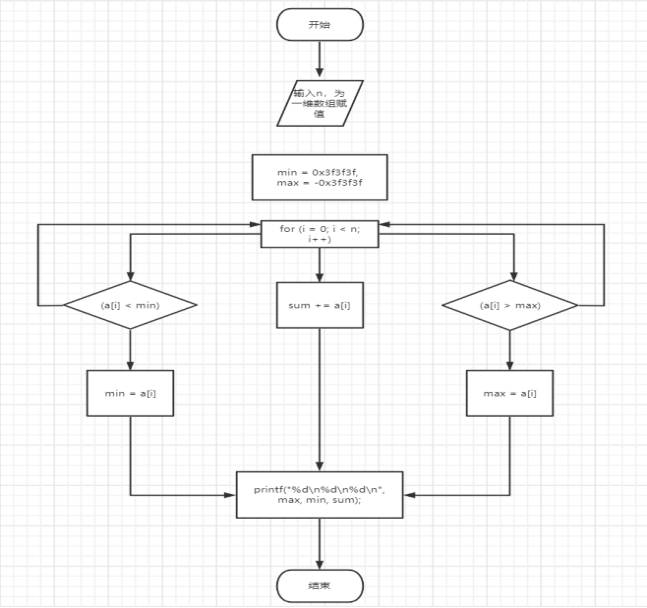


**3、【BASIC-4】数列特征**

给出n个数，找出这n个数的最大值，最小值，和。

**算法思路: 边读入边处理最大值最小值和求和,最后输出即可**

**流程图:**



**源代码:**

#include "stdio.h"

int main()

{

int i, t, n, max, min, sum;

scanf("%d%d", &n, &max);

sum = min = max;

for (i = 1; i < n; ++i) {

scanf("%d", &t);

if (max < t) max = t;

if (min > t) min = t;

sum += t;

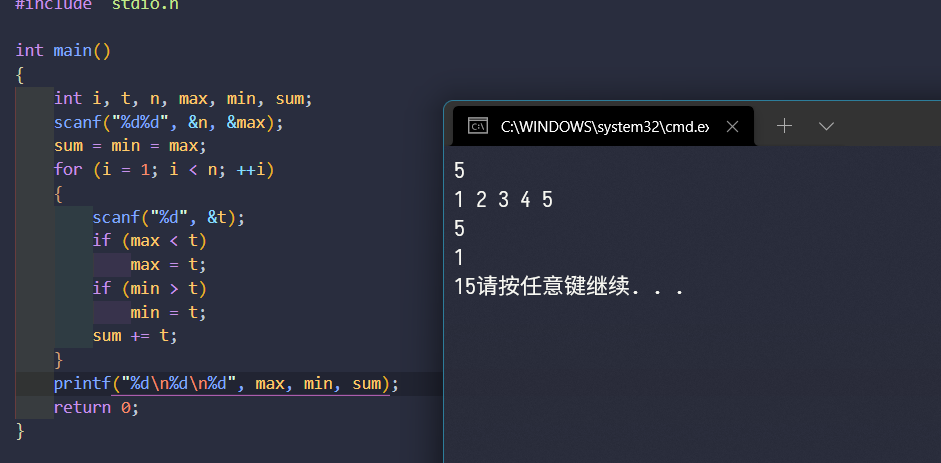
}

printf("%d\n%d\n%d", max, min, sum);

return 0;

}

**运行截图:**



**提交记录:**

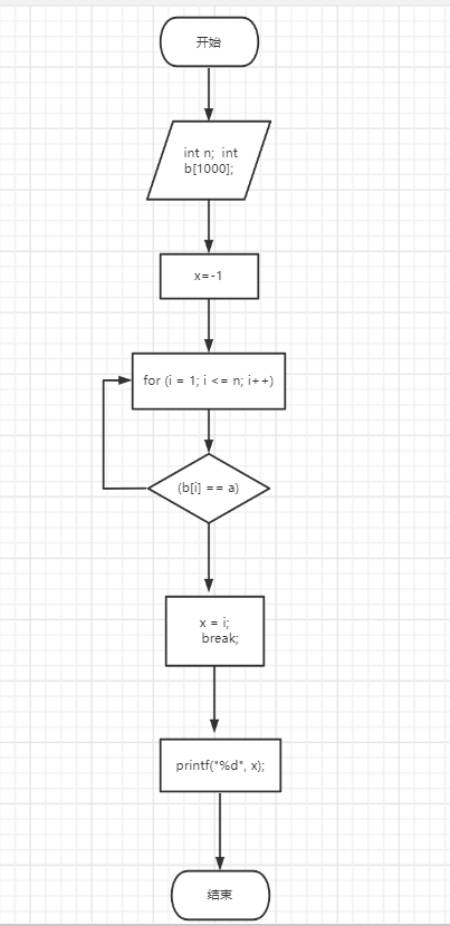


**4、【BASIC-5】查找整数**

给出一个包含n个整数的数列，问整数a在数列中的第一次出现是第几个。

**算法思路: for循环读入所有数,之后再此遍历判断读入的数是否是目标a,如果是则输出,否则继续判断,如果都不是则输出-1**

**流程图:**



**源代码:**

#include "stdio.h"

int main()

{

int i, n, t, flag, arr[1001];

flag = 0;

scanf("%d", &n);

for (i = 0; i < n; ++i)

scanf("%d", &arr[i]);

scanf("%d", &t);

for (i = 0; i < n; ++i){

if (arr[i] == t){

printf("%d", i + 1);

flag = 1;

break;

}

}

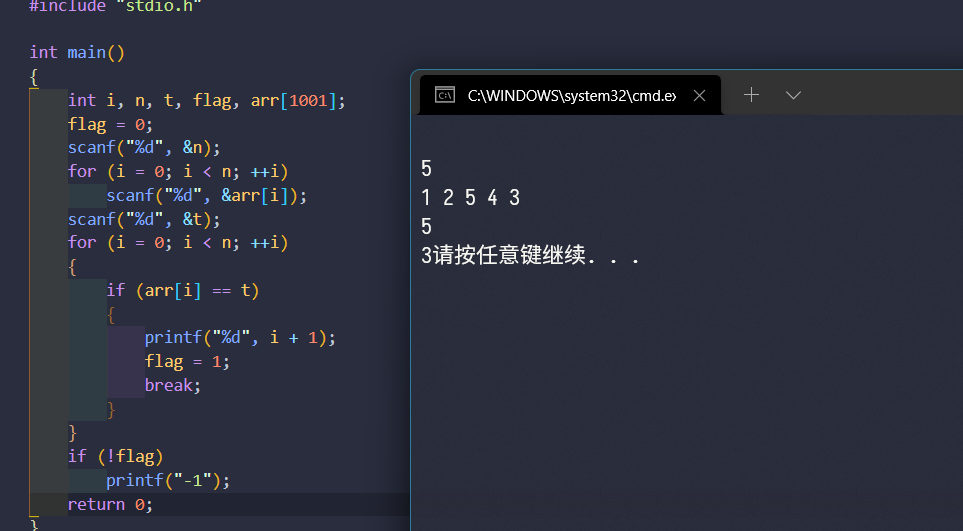
if (!flag)

printf("-1");

return 0;

}

**运行截图:**



**提交记录:**



**5、【BASIC-6】杨辉三角形**

杨辉三角形又称Pascal三角形，它的第i+1行是(a+b)i的展开式的系数。它的一个重要性质是：三角形中的每个数字等于它两肩上的数字相加。下面给出了杨辉三角形的前4行：

1

1 1

1 2 1

1 3 3 1

给出n，输出它的前n行。

**算法思路:**根据杨辉三角的形式，不难看出是会用到二维数组,杨辉三角的公式：a[i][j]=a[i-1][j-1]+a[i-1][j]，即下面的数字等于肩上两个数字的和。

**流程图:**

**源代码:**

#include "stdio.h"

int main()

{

int i, j,n;

scanf("%d", &n);

int a[n][n];

for (i = 0; i < n; ++i) {

for (j = 0; j < i + 1; ++j) {

if (i == j || j == 0)

a[i][j] = 1;

if (i > 1 && j > 0 && i != j)

a[i][j] = a[i - 1][j - 1] + a[i - 1][j];

printf("%d ",a[i][j]);

}

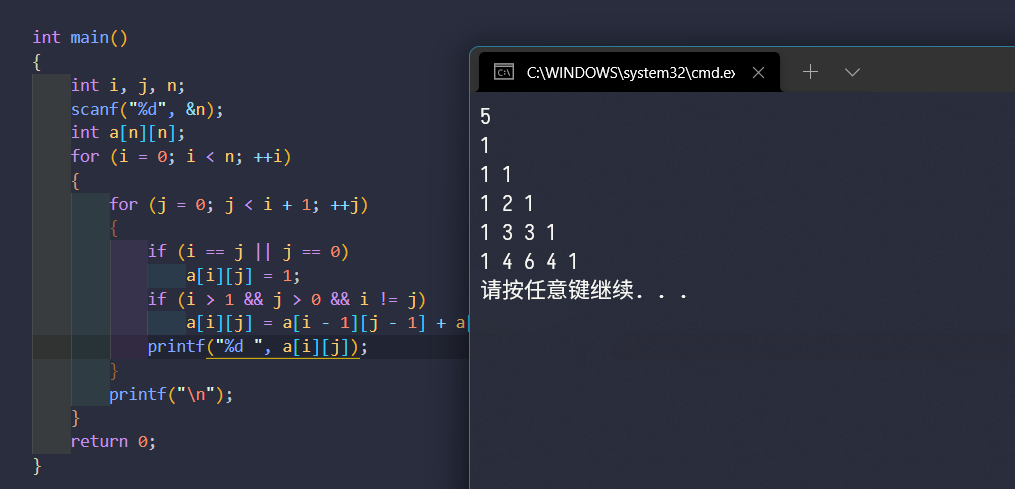
printf("\n");

}

return 0;

}

**运行截图:**



**提交记录:**

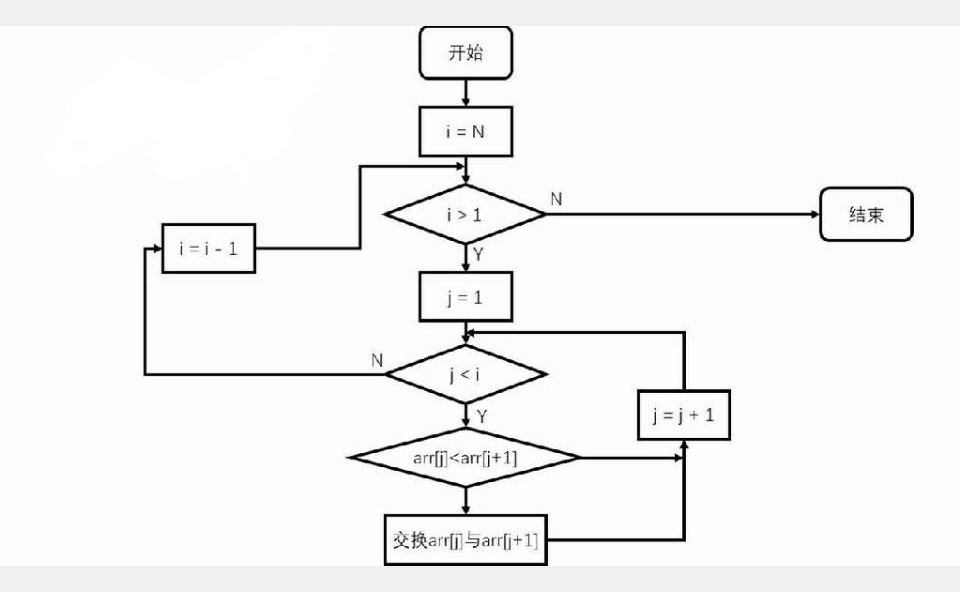


**6、【BASIC-13】数列排序**

给定一个长度为n的数列，将这个数列按从小到大的顺序排列。1<=n<=200

**算法思路: 给数组排序即可**

**流程图:**

****

**源代码:**

#include "stdio.h"

#include "stdlib.h"

#include"string.h"

#include "math.h"

void f(int\* a, int len) {

int i, j, t;

for (i = 0; i < len; i++) {

for (j = i + 1; j < len; j++) {

if (a[i] > a[j]) {

t = a[i];

a[i] = a[j];

a[j] = t;

}

}

}

for (i = 0; i < len; i++) {

if (i == len - 1)

printf("%d", a[i]);

else

printf("%d ", a[i]);

}

}

int main()

{

int i, n, a[200];

scanf("%d", &n);

for (i = 0; i < n; ++i) {

scanf("%d", &a[i]);

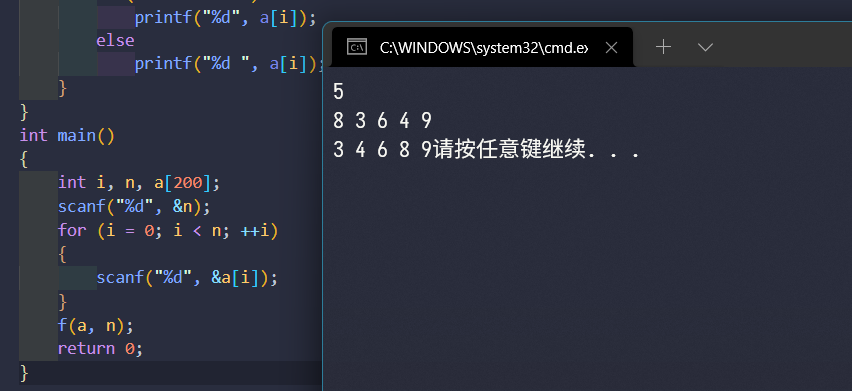
}

f(a, n);

return 0;

}

**运行截图:**



**提交记录:**



1. 算法训练

**7、【ALGO-955】P0701**

编写一个函数RegularPlural，其功能是实现一个英文单词的复数形式。复数的规则为：

（1） 如果单词末尾为s,x,z,ch或sh，则在后面加es

（2） 如果单词末尾为y，且前一个字母为辅音（除a, e, i, o, u以外的其它情况），则把y改成ies。

（3） 如果是其它情形，一律在后面加s。

编写测试程序，输入一个长度小于20的单词，输出该单词的复数形式。

**算法思路: 根据题目条件进行ifelse判断写出代码**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

string s;

void solve()

{

int len = s.length();

string copy = s;

if (s[len - 1] == 's' || s[len - 1] == 'x' || s[len - 1] == 'z')

{

s += "es";

}

else if (len >= 2)

{

{

if (s.substr(len - 2, 2) == "ch" || s.substr(len - 2, 2) == "sh")

{

s += "es";

}

if (s[len - 1] == 'y')

{

if (s[len - 2] != 'a' && s[len - 2] != 'e' && s[len - 2] != 'i' && s[len - 2] != 'o' && s[len - 2] != 'u')

{

s[len - 1] = 'i';

s += "es";

}else{

s += 's';

}

}

}

}

if(copy == s)

s += 's';

cout << s;

}

int main()

{

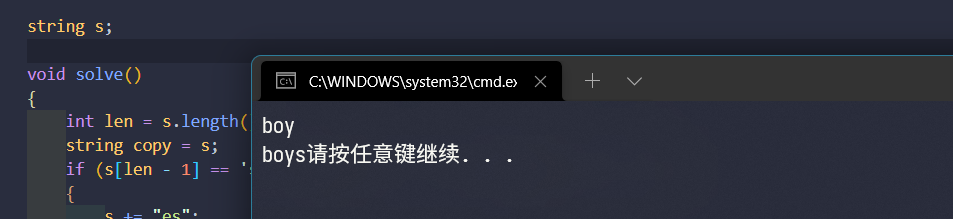
cin >> s;

solve();

return 0;

}

**运行截图:**



**提交截图:**



**8、【ALGO-956】P0702**

在C语言中，有一个strcmp函数，其功能是比较两个字符串s1和s2。请编写一个你自己的字符串比较函数my\_strcmp，来实现strcmp函数的类似功能。如果s1=s2，则返回0；否则返回s1 与s2 第一个不同字符的差值（如果s1<s2，该差值是一个负数；如果s1>s2，该差值是一个正数）。编写测试程序，输入两个长度小于1000的字符串（可能包含有空格，且长度不一定相等），然后调用my\_strcmp函数来进行比较，并输出返回结果。

**算法思路:for循环比较相同位,如果s1.size() == s2.size()当发现不同时,返回**s1[i] - s2[i]; 如果**s1.size() != s2.size(), 如果s1.size() < s2.size(),返回-s2[i],如果s1.size() > s2.size(),返回s1[i]**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

string s1, s2;

int my\_strcmp(string s1, string s2)

{

int i = 0;

while (i < s1.size() && i < s2.size())

{

if (s1[i] != s2[i])

return s1[i] - s2[i];

else

i++;

}

if (i < s1.size())

return s1[i];

else

return -s2[i];

}

int main()

{

getline(cin, s1);

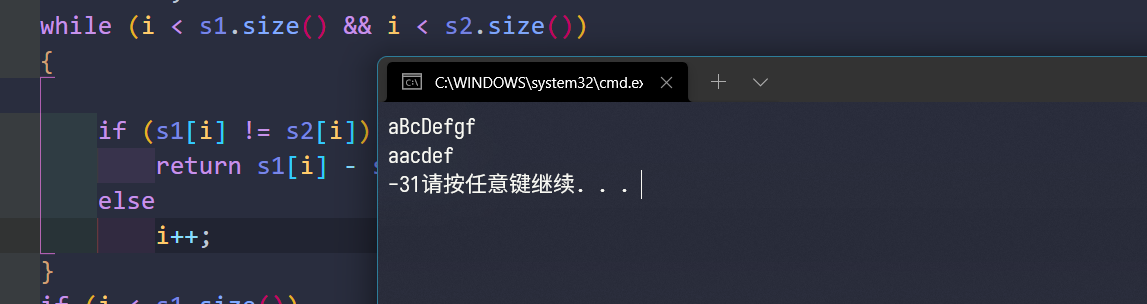
getline(cin, s2);

cout << my\_strcmp(s1, s2);

return 0;

}

**运行截图:**



**提交截图:**



**9、【ALGO-957】P0703**

一个整数的反置数指的是把该整数的每一位数字的顺序颠倒过来所得到的另一个整数。如果一个整数的末尾是以0结尾，那么在它的反置数当中，这些0就被省略掉了。比如说，1245的反置数5421，而1200的反置数是21。请编写一个程序，输入两个整数，然后计算这两个整数的反置数之和sum，然后再把sum的反置数打印出来。例如，如果用户输入：435和754，则输出结果为199。要求：由于在本题中需要多次去计算一个整数的反置数，因此必须把这部分代码抽象为一个函数的形式。

**算法思路:（1）先用cnt计算出输入数字位数**

**（2）对数字取余提取出各位数，反置**

**（3）主函数调用反置函数求和**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

string s1, s2;

void solve(string s1, string s2){

int a, b;

a = b = 0;

for (int i = s1.size() - 1; i >= 0; i--){

a = a \* 10 + (s1[i] - '0');

}

for (int i = s2.size() - 1; i >= 0; i--)

{

b = b \* 10 + (s2[i] - '0');

}

int sum = a + b;

while(sum % 10 == 0){

sum /= 10;

}

while(sum){

cout << sum % 10;

sum /= 10;

}

}

int main()

{

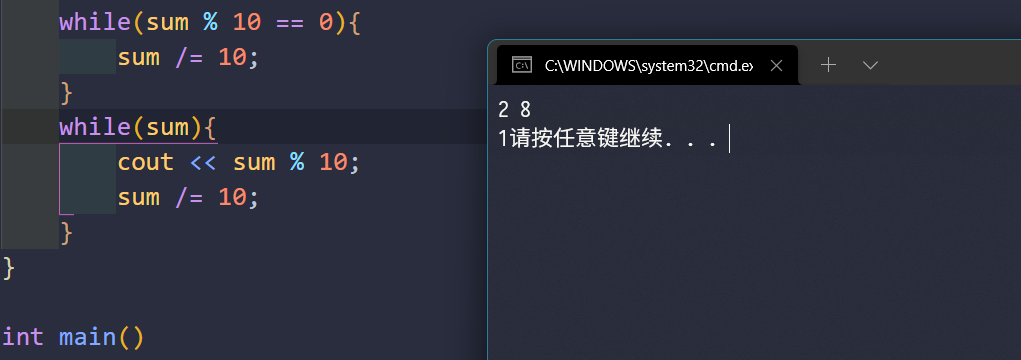
cin >> s1 >> s2;

solve(s1, s2);

return 0;

}

**运行截图:**



**提交截图:**



**10、【ALGO-958】P0704**

一个数如果从左往右读和从右往左读数字是完全相同的，则称这个数为回文数，比如898,1221,15651都是回文数。编写一个程序，输入两个整数min和max，然后对于min~max之间的每一个整数（包括min和max），如果它既是一个回文数又是一个质数，那么就把它打印出来。要求，回文数和质数的判断都必要要用函数的形式来实现。

**算法思路:首先找出范围内的素数,如果是素数,则判断是否是回文,如果也是回文则输出**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 1e6 + 10;

int a, b;

int vis[N];

void prime(int N )

{

for (int i = 2; i <= N - 1; i++)

vis[i] = 1;

for (int i = 2; i <= N - 1 ; i++)

{

if (vis[i])

{

for (int j = i << 1; j <= N -1 ; j += i)

{

vis[j] = 0;

}

}

}

}

int pdhw(int x)

{

int arr[20], cnt = 0;

int i, j;

memset(arr, 0, sizeof(arr));

while (x)

{

arr[cnt++] = x % 10;

x /= 10;

}

i = 0, j = cnt - 1;

while (i < j)

{

if (arr[i] != arr[j])

return 0;

else

i++, j--;

}

return 1;

}

int main()

{

cin >> a >> b;

prime(N);

int flag = 0;

for (int i = a; i <= b; i++)

{

if (vis[i])

{

if (pdhw(i)){

if(!flag)

cout << i, flag = 1;

else

cout << " " << i;

}

}

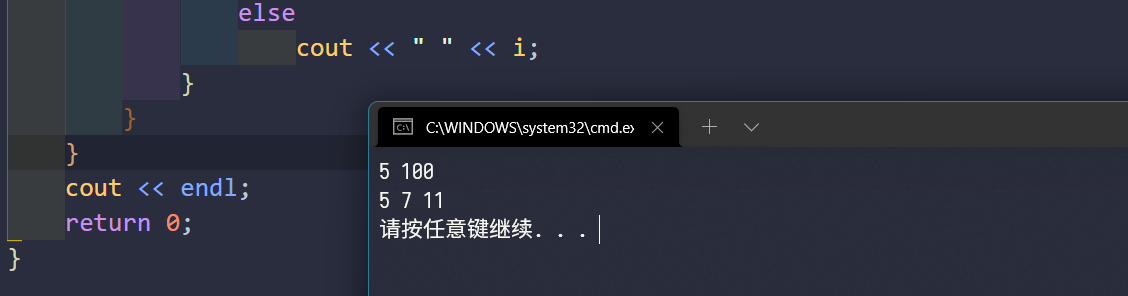
}

cout << endl;

return 0;

}

**运行截图:**



**提交截图:**



**11、【ALGO-941】P0601**

编写一个程序，先输入一个字符串str（长度不超过20），再输入单独的一个字符ch，然后程序会把字符串str当中出现的所有的ch字符都删掉，从而得到一个新的字符串str2，然后把这个字符串打印出来。

**算法思路:利用函数**replace 将所有的ch替换成空即可

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 1e6 + 10;

string s, t;

int main()

{

cin >> s >> t;

while(s.find(t) != s.npos){

s = s.replace(s.find(t), 1, "");

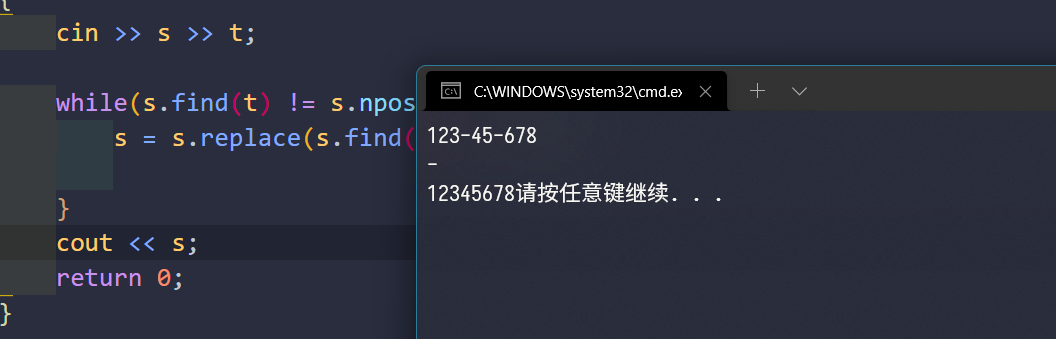
}

cout << s;

return 0;

}

**运行截图:**



**提交截图:**

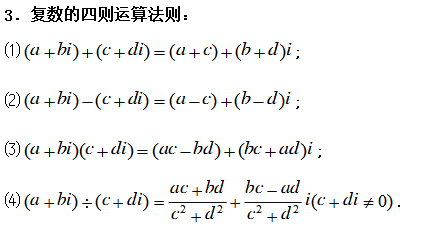


**12、【ALGO-142】P1103**

编程实现两个复数的运算。

要求（1）定义一个结构体类型来描述复数。

　　（2）复数之间的加法、减法、乘法和除法分别用不用的函数来实现。



**算法思路: 直接模拟复数的四则运算即可**

**流程图:**

**源代码:**

#include<stdio.h>

typedef struct complex{

double x;

double y;

} f;

void fushu(f \* fs){

scanf("%lf%lf",&fs->x,&fs->y);

}

void jia(f f1,f f2,f \*w){

w->x=f1.x+f2.x;

w->y=f1.y+f2.y;

}

void jian(f f1,f f2,f \*w){

w->x=f1.x-f2.x;

w->y=f1.y-f2.y;

}

void cheng(f f1,f f2,f \*w){

w->x = f1.x \* f2.x - f1.y \* f2.y;

w->y = f1.y \* f2.x + f1.x \* f2.y;

}

void chu(f f1,f f2,f \*w){

w->x = ( f1.x \* f2.x + f1.y \* f2.y ) / ( ( f2.x \* f2.x) + ( f2.y \* f2.y) );

w->y = ( f1.y \* f2.x - f1.x \* f2.y ) / ( ( f2.x \* f2.x) + ( f2.y \* f2.y) );

}

int main(){

f f1 ;

f f2 ;

f jg ;

char fh ;

scanf("%c", &fh );

fushu(&f1);

fushu(&f2);

switch(fh){

case '+':

jia(f1,f2,&jg);

break;

case '-':

jian(f1,f2,&jg);

break;

case '\*':

cheng(f1,f2,&jg);

break;

case '/':

chu(f1,f2,&jg);

break;

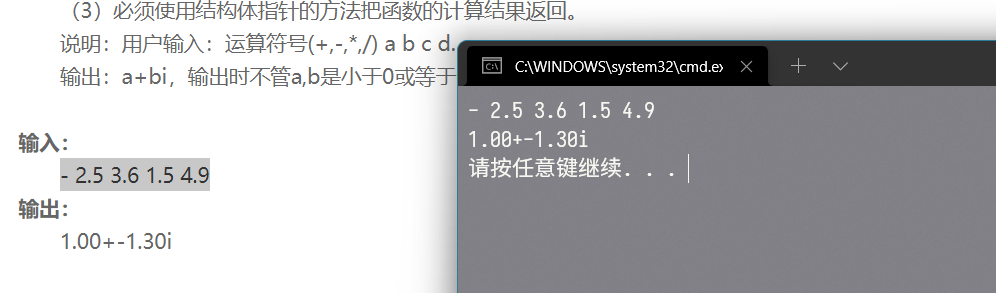
}

printf("%.2lf+%.2lfi\n", jg.x , jg.y );

return 0;

}

**运行截图:**



**提交截图:**



**13、【ALGO-79】删除数组零元素**

从键盘读入n个整数放入数组中，编写函数CompactIntegers，删除数组中所有值为0的元素，其后元素向数组首端移动。注意，CompactIntegers函数需要接受数组及其元素个数作为参数，函数返回值应为删除操作执行后数组的新元素个数。输出删除后数组中元素的个数并依次输出数组元素。

**算法思路:**

1，接收用户输入的数据，

2，存储不为0的出现次数

3，遇到0就continue，否则就输出

**流程图:**

**源代码:**

#include <iostream>

using namespace std;

int main()

{

    int n, i, m = 0, j = 0;

    cin >> n;

    int a[n + 8];

    int b[n + 8];

    for (i = 0; i < n; i++)

    {

        cin >> a[i];

        if (a[i] != 0)

        {

            m++;

            b[j++] = a[i];

        }

    }

    if (m == 0)

    {

        cout << 0;

    }

    else

    {

        cout << m << endl;

        for (i = 0; i < m; i++)

        {

            cout << b[i] << " ";

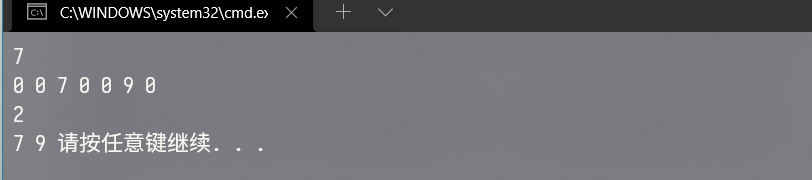
        }

    }

    return 0;

}

**运行截图:**



**提交截图:**



**14、【ALGO-467】大整数加法**

任意输入两个正整数a,b，求两数之和。（注：本题会输入超过32位整数限制的大整数）

建议使用字符数组实现。

**算法思路:**用数组模拟加法过程即可。

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 100;

string a, b;

vector<int> A, B, C;

vector<int> add(vector<int> A, vector<int> B){

int t = 0;

for (int i = 0; i < A.size() || i < B.size(); i++){

if(i < A.size()) t += A[i];

if(i < B.size()) t += B[i];

C.push\_back(t % 10);

t /= 10;

}

if(t) C.push\_back(1);

return C;

}

int main()

{

cin >> a >> b;

for(int i = a.size() - 1; i >= 0; i--) A.push\_back(a[i] - '0');

for(int i = b.size() - 1; i >= 0; i--) B.push\_back(b[i] - '0');

C = add(A, B);

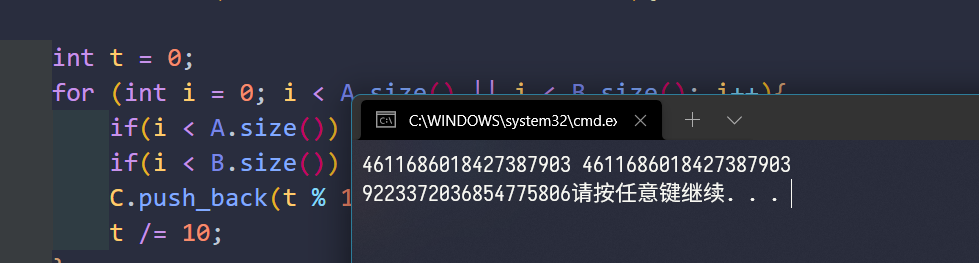
for(int i = C.size() - 1; i >= 0; i--)

cout << C[i];

return 0;

}

**运行截图:**



**提交截图:**



**15、【ALGO-53】最小乘积(基本型)**

给两组数，各n个。

请调整每组数的排列顺序，使得两组数据相同下标元素对应相乘，然后相加的和最小。要求程序输出这个最小值。

例如两组数分别为:1 3 -5和-2 4 1

　　那么对应乘积取和的最小值应为：

(-5) \* 4 + 3 \* (-2) + 1 \* 1 = -25

**算法思路: 对a数组升序排列,b数组降序排列,然后分别相乘相加输出即可**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 100;

int a[N], b[N], t, n;

string s, s1, s2;

int ans, res, sum;

int main()

{

cin >> t;

while(t --){

cin >> n;

memset(a, 0, sizeof a);

memset(b, 0, sizeof b);

for (int i = 0; i < n; i ++) cin >> a[i];

for (int i = 0; i < n; i ++) cin >> b[i];

sort(a, a + n);

sort(b, b + n, greater<int>());

sum = 0;

for (int i = 0; i < n; i++) {

sum += a[i] \* b[i];

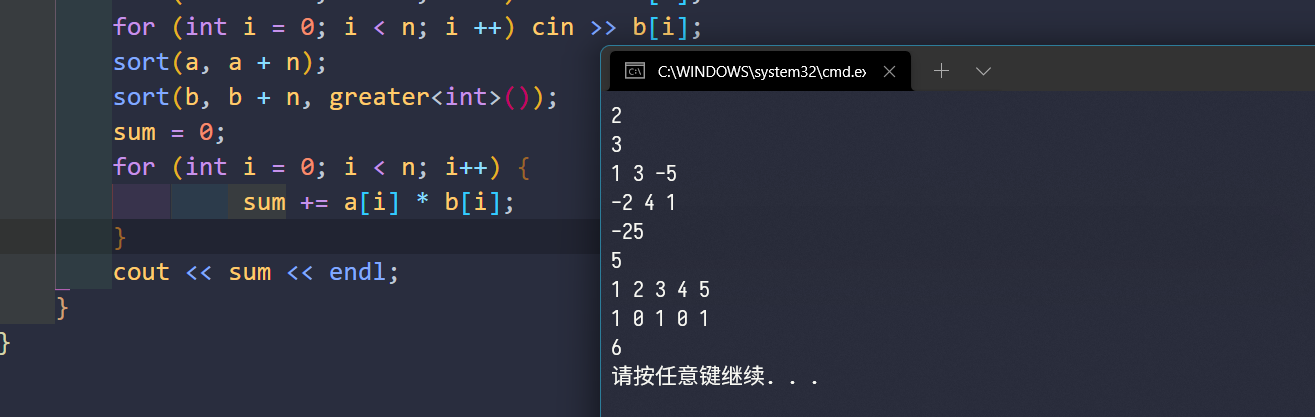
}

cout << sum << endl;

}

}

**运行截图:**



**提交截图:**



**16、【ALGO-561】矩阵运算**

给定一个n\*n的矩阵A，求A+AT的值。其中AT表示A的转置。

**算法思路:**使用矩阵乘法公式，用二维数组模拟矩阵。

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 110;

int a[N], b[N], arr[N][N], t, n;

string s, s1, s2;

int ans, res, sum;

int main()

{

cin >> n;

int copy[N][N];

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++){

cin >> arr[i][j];

copy[i][j] = arr[i][j];

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

copy[i][j] += arr[j][i];

cout << copy[i][j] << ' ';

}

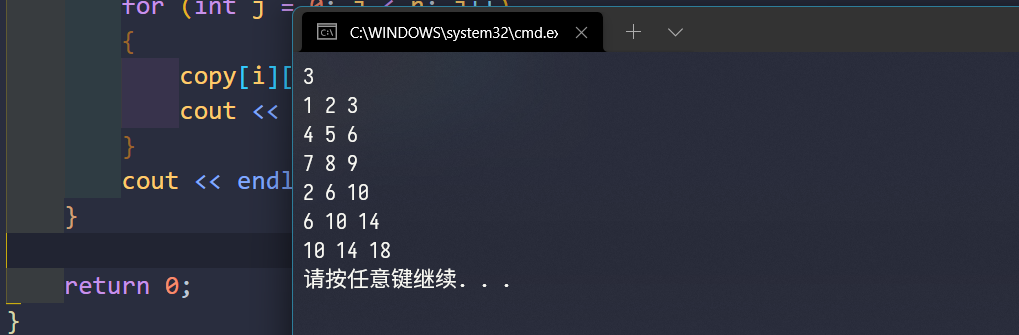
cout << endl;

}

return 0;

}

**运行截图:**



**提交截图:**



**17、【ALGO-551】百鸡百钱**

我国古代数学家张丘建在《算经》一书中提出的数学问题：鸡翁一值钱五，鸡母一值钱三，鸡雏三值钱一。百钱买百鸡，问鸡翁、鸡母、鸡雏各几何？

**算法思路: 3层for循环分别枚举鸡翁,鸡母,鸡雏的数量,如果符合(钱数==100 && 个数 == 100) 则输出**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 110;

int a[N], b[N], arr[N][N], t, n;

string s, s1, s2;

int ans, res, sum;

int main()

{

for (int i = 0; i <= 20; i ++){

for (int j = 0; j <= 33; j ++){

for (int k = 0; k <= 300; k += 3){

if(i\*5 + j\*3 + k/3 == 100 && i + j + k == 100){

cout << i << " " << j << " " << k << endl;

}

}

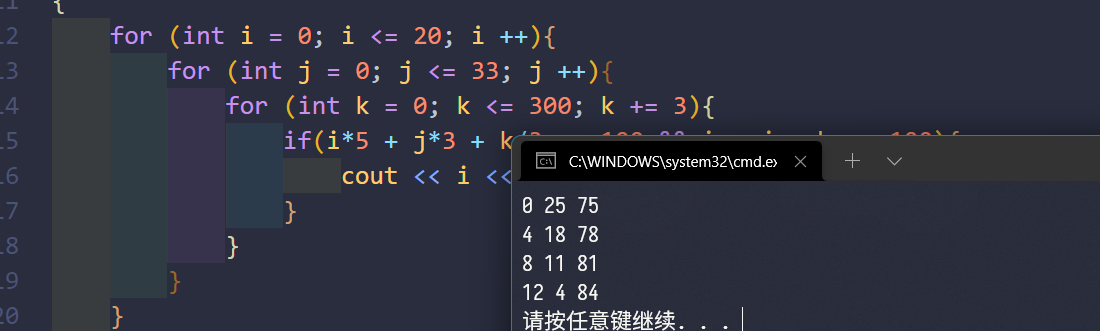
}

}

return 0;

}

**运行截图:**



**提交截图:**



**18、【ALGO-538】数据传递加密**

某个公司传递数据，数据是四位整数，在传递过程中需要进行加密的，加密规则如下：每位数字都加上5,然后除以10的余数代替该位数字。再将新生成数据的第一位和第四位交换，第二位和第三位交换。要求输入4位整数，输出加密后的4位整数。比如：输入一个四位整数1234，则输出加密结果为9876。

**算法思路: 根据题意模拟编写代码即可**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 110;

int a[N], b[N], arr[N][N], t, n;

string s, s1, s2;

int ans, res, sum;

int main()

{

cin >> s;

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

int re = (s[i] - '0' + 5) % 10;

s[i] = re + '0';

}

cout << s[3] << s[2] << s[1] << s[0];

return 0;

}

**运行截图:**



**提交截图:**



1. 算法提高

**19、【ADV-1082】心形**

根据给出的最大宽度，输出心形（注意空格，行末不加多余空格）。

**算法思路:**寻找规律。将心形拆分成上下两个部分。上半部分将第一行拆 分出来考虑。以最大宽度行作为基准行，心形上半部分有s=n/4行，第 一行有2s+1个空格，除第一行外上部分每行前面空格为 2s-1,2s-3,2s-5....;中间含有空格数量为3+4（s-1），3+4（s-2）， 3+4（s-3）.....；以“\* \* ”为一个组合，每行两侧分别有1,2,3.... 组；  
 心形下半部分，有2（s-3）+7；每行前面空格数为1,3,5,7,9....依次 增加2。以“\* ”为一组，每往下一行减少一组，最大宽度行有n-1组， 依次往下减少一组，即为n-2，n-3...；每行单独补上一个“\*”。

**流程图:**

**源代码:**

#include <stdio.h>

int main(){

int x,i,j;

int control,control1=1,control2=1;

scanf("%d",&x);

// 控制每一列

for(i=1;i<x/4+x/2+2;i++){ //i < 13/2/2 + 13/2 + 1 + 1

// 上半部分

if(i<=x/4){

control1 = control2=1;

// 控制每一行

for(j=1;j<2\*x+1;j++){ // j<26+1 (x-10- i)\*2+ 1 < (x- 10-i)\*2+1

// 画\*的区域

if(j>x/2-2\*i+3 && j<x/2+2\*i+1){ // 13/2-2\*1 + 3<j<13/2 + 2\*1 7<j<9; 5< j< 11

printf("\* ");

j++;

}

else if(j>x/2+x-2\*i+2 && j<x/2+x+2\*i+1){ // 13/2 + 13 - 2\*i+2 < j< 13/2 + 13 + 2\*i + 1 19<j<22

printf("\* ");

j++;

}

else printf(" ");

}

}

// 下半部分

else{

control=1;

for(j=1;j<2\*x+2;j++){

if(j>2\*i-x/2-1 && j<2\*x+x/2-2\*i+3){

printf("\* ");

j++;

}

else printf(" ");

}

}

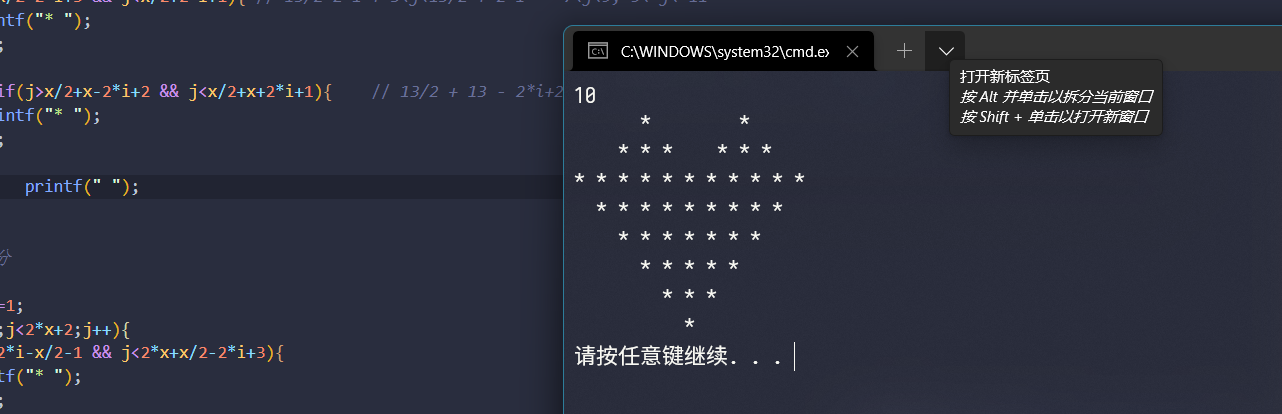
printf("\n");

}

return 0;

}

**运行截图:**



**提交截图:**



**20、【ADV-658】字符串查找**

给定两个字符串a和b，查找b在a中第一次出现的位置。

如a=“hello world”，b=“world”时，b第一次出现是在a的第7个字符到第11个字符，按照C++的习惯，位置从0开始编号，所以b在a中第一次出现的位置为6。

**算法思路: 从左往右利用字符串函数substr取出子串比较,若相等,则输出i,break;**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 110;

int a[N], b[N], arr[N][N], t, n;

string s, s1, s2;

int ans, res, sum;

int main()

{

    getline(cin, s);

    getline(cin, s1);

    for (int i = 0; i < s.size() - s1.size(); i++)

    {

        if (s.substr(i, s1.size()) == s1)

        {

            cout << i << endl;

            return 0;

        }

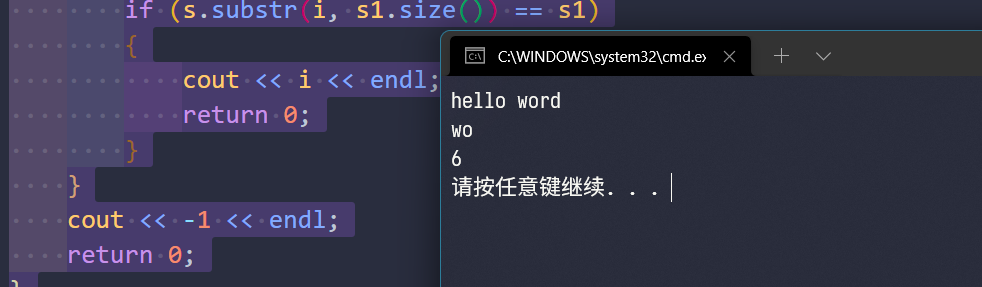
    }

    cout << -1 << endl;

    return 0;

}

**运行截图:**



**提交截图:**



**21、【ADV-789】校门外的树**

某校大门外长度为L的马路上有一排树，每两棵相邻的树之间的间隔都是1米。我们可以把马路看成一个数轴，马路的一端在数轴0的位置，另一端在L的位置；数 轴上的每个整数点，即0，1，2，……，L，都种有一棵树。

由于马路上有一些区域要用来建地铁。这些区域用它们在数轴上的起始点和终止点表示。已知任一区域的起始点和终止点的坐标都是整数，区域之间可能有重合的部分。现在要把这些区域中的树（包括区域端点处的两棵树）移走。你的任务是计算将这些树 都移走后，马路上还有多少棵树。

**算法思路:利用区间合并将小区间合成为一个一个的大区间,最后计算即可**

**流程图:**

**源代码:**

#include <algorithm>

#include <iostream>

#include <vector>

using namespace std;

const int N = 100010;

typedef pair<int, int> PII;

vector<PII> segs;

void merge(vector<PII> &segs)

{

vector<PII> res;

sort(segs.begin(), segs.end());

int st = -2e9, ed = -2e9;

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

{

PII seg = segs[i];

if (ed < seg.first)

{

if (st != -2e9)

res.push\_back({st, ed});

st = seg.first, ed = seg.second;

}

else

ed = max(ed, seg.second);

}

if (st != -2e9)

res.push\_back({st, ed});

segs = res;

}

int main()

{

int L, m;

cin >> L >> m;

for (int i = 0; i < m; i++)

{

int l, r;

cin >> l >> r;

segs.push\_back({l, r});

}

merge(segs);

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

{

L -= (segs[i].second - segs[i].first + 1);

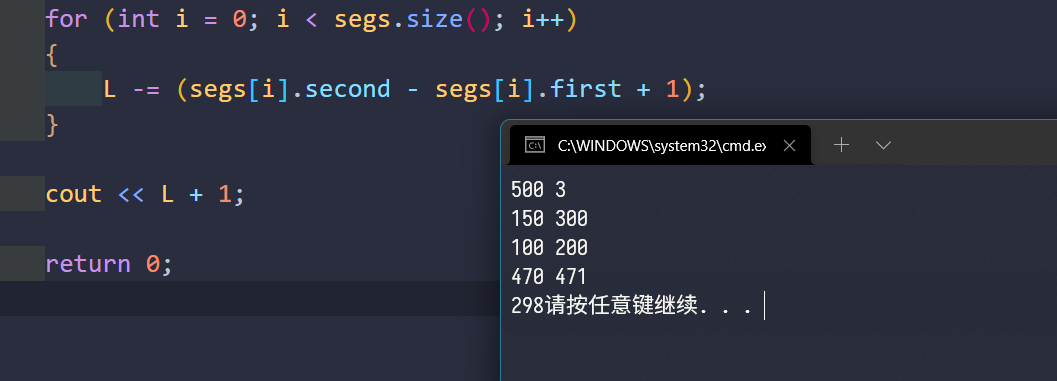
}

cout << L + 1;

return 0;

}

**运行截图:**



**提交截图:**



**22、【ADV-100】第二大整数**

编写一个程序，读入一组整数（不超过20个），当用户输入0时，表示输入结束。然后程序将从这组整数中，把第二大的那个整数找出来，并把它打印出来。说明：（1）0表示输入结束，它本身并不计入这组整数中。（2）在这组整数中，既有正数，也可能有负数。（3）这组整数的个数不少于2个。

**算法思路: 读入所有元素将其放入大根堆,读取完毕后移除堆顶元素,更新后的堆顶即为第二大数**

**流程图:**

**源代码:**

#include <algorithm>

#include <iostream>

#include <vector>

using namespace std;

const int N = 100010;

typedef pair<int, int> PII;

vector<int> v;

int main()

{

int x;

while (cin >> x)

{

if(x == 0)

break;

else{

v.push\_back(x);

}

}

make\_heap(v.begin(), v.end(), less<int>());

pop\_heap(v.begin(), v.end());

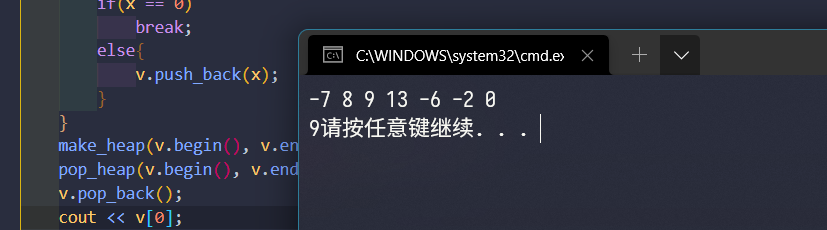
v.pop\_back();

cout << v[0];

return 0;

}

**运行截图:**



**提交截图:**



1. 历届试题

**23、【PREV-151】四平方和**

四平方和定理，又称为拉格朗日定理：每个正整数都可以表示为至多4个正整数的平方和。　　如果把0包括进去，就正好可以表示为4个数的平方和。

　　比如：

　　5 = 0^2 + 0^2 + 1^2 + 2^2

　　7 = 1^2 + 1^2 + 1^2 + 2^2

　　（^符号表示乘方的意思）

　　对于一个给定的正整数，可能存在多种平方和的表示法。

　　要求你对4个数排序：

　　0 <= a <= b <= c <= d

　　并对所有的可能表示法按 a,b,c,d 为联合主键升序排列，最后输出第一个表示法

**算法思路:先枚举cd的值，然后将它们可以组成的数存到数组里。之后在，枚举ab，那么只要数组中存在n-a\*a-b\*b就是答案，输出。**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 6000000;

int n;

int c[N], d[N];

int main()

{

cin >> n;

memset(c, -1, sizeof(c));

for (int \_c = 0; \_c \* \_c <= n; \_c++){

for (int \_d = \_c; \_d \* \_d + \_c \* \_c <= n; \_d++){

int t = \_c\*\_c + \_d \* \_d;

if(c[t] == -1){

c[t] = \_c;

d[t] = \_d;

}

}

}

for(int \_a = 0; \_a \* \_a <= n; \_a++){

for (int \_b = \_a; \_b \* \_b + \_a \* \_a <= n; \_b++)

{

if (c[n - \_a \* \_a - \_b \* \_b] != -1)

{

cout << \_a << ' ' << \_b << ' ' << c[n - \_a \* \_a - \_b \* \_b] << ' ' << d[n - \_a \* \_a - \_b \* \_b];

return 0;

}

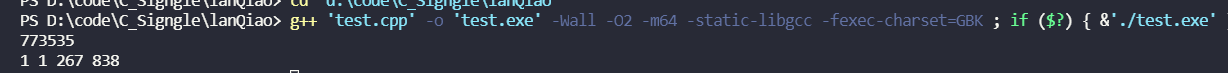
}

}

return 0;

}

**运行截图:**



**提交截图:**



**24、【PREV-286】错误票据**

某涉密单位下发了某种票据，并要在年终全部收回。每张票据有唯一的ID号。全年所有票据的ID号是连续的，但ID的开始数码是随机选定的。因为工作人员疏忽，在录入ID号的时候发生了一处错误，造成了某个ID断号，另外一个ID重号。你的任务是通过编程，找出断号的ID和重号的ID。假设断号不可能发生在最大和最小号。

**算法思路: map[i]表示数字i出现的次数,同时用minn和maxx表示连续的区间最小最大值,最后循环找出map[i](minn <= i <= maxx)的值为0和1的i输出即可**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 1000;

map<int, int> mp;

int n;

int minn = 99999999, maxx = -1;

int main()

{

cin >> n;

while (n --){

int x;

while(cin >> x){

minn = min(minn,x);

maxx = max(maxx, x);

mp[x] += 1;

}

}

int res1, res2;

for (int i = minn; i <= maxx; i ++)

{

if(mp[i] == 0){

res1 = i;

}

if(mp[i] == 2){

res2 = i;

}

}

cout << res1 << ' ' << res2;

return 0;

}

**运行截图:**



**提交截图:**



**25、【PREV-343】密码发生器**

在对银行账户等重要权限设置密码的时候，我们常常遇到这样的烦恼：如果为了好记用生日吧，容易被破解，不安全；如果设置不好记的密码，又担心自己也会忘记；如果写在纸上，担心纸张被别人发现或弄丢了。

这个程序的任务就是把一串拼音字母转换为6位数字（密码）。我们可以使用任何好记的拼音串(比如名字，王喜明，就写：wangximing)作为输入，程序输出6位数字。

　　变换的过程如下：

第一步. 把字符串6个一组折叠起来，比如wangximing则变为：

　　 wangxi

　 Ming代码

　　第二步. 把所有垂直在同一个位置的字符的ascii码值相加，得出6个数字，如上面的例子， 则得出：

　　228 202 220 206 120 105

　　第三步. 再把每个数字“缩位”处理：就是把每个位的数字相加，得出的数字如果不是一位数字， 就再缩位，直到变成一位数字为止。例如: 228 => 2+2+8=12 => 1+2=3

　　上面的数字缩位后变为：344836, 这就是程序最终的输出结果！

**算法思路: 根据题意写出代码即可**

**流程图:**

**源代码:**

#include <bits/stdc++.h>

using namespace std;

const int N = 6000000;

int n;

string s;

int res[10];

int main()

{

cin >> n;

while (n --)

{

memset(res, 0, sizeof res);

cin >> s;

for(int i = 0; i < 6; i++){

int t = i;

while (t < s.size()){

res[i] += s[t];

t += 6;

}

}

// for(int i = 0; i < 6; i++) cout << res[i] << endl;

for (int i = 0; i < 6; i++){

int t;

int x = res[i];

do{

t = 0;

while (x)

{

t += x % 10;

x /= 10;

}

x = t;

}while(t >= 10);

res[i] = t;

}

for (int i = 0; i < 6; i++) cout << res[i];

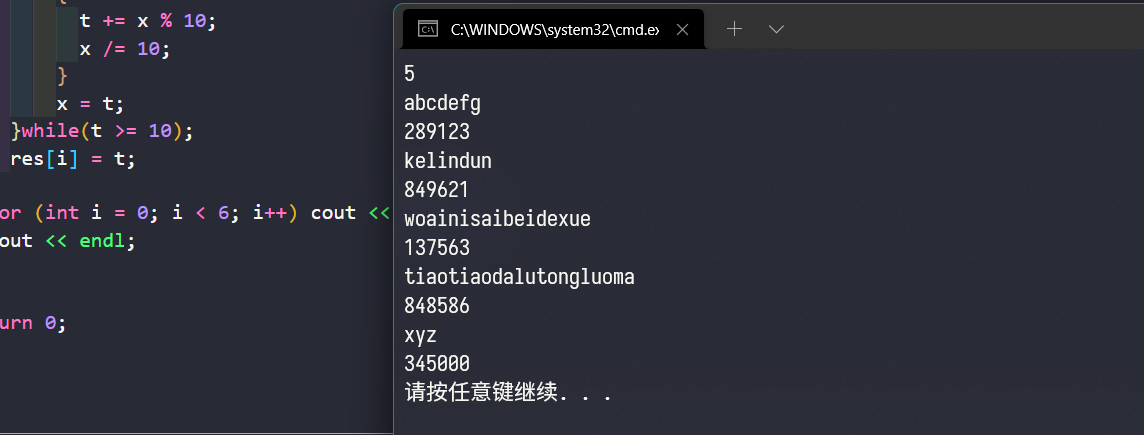
cout << endl;

}

return 0;

}

**运行截图:**



**提交截图:**

