

**7-1 计算存款利息** 本题目要求计算存款利息，计算公式为interest=money×(1+rate)​year​​−money，其中interest为存款到期时的利息（税前），money是存款金额，year是存期，rate是年利率。

输入格式：输入在一行中顺序给出三个正实数money、year和rate，以空格分隔。

输出格式：在一行中按“interest = 利息”的格式输出，其中利息保留两位小数。

输入样例：1000 3 0.025

输出样例 interest = 76.89

**#include <iostream>**

**#include <math.h>**

**using namespace std;**

**int main(){**

**float money, year;**

**float rate;**

**cin >> money >> year >> rate;**

**float interest = money \* pow((1+rate),year) - money;**

**cout << "interest = " << round(interest\*100)/100 << endl;**

**return 0;**

**}**

**7-2 然后是几点** 有时候人们用四位数字表示一个时间，比如1106表示11点零6分。现在，你的程序要根据起始时间和流逝的时间计算出终止时间。

读入两个数字，第一个数字以这样的四位数字表示当前时间，第二个数字表示分钟数，计算当前时间经过那么多分钟后是几点，结果也表示为四位数字。当小时为个位数时，没有前导的零，即5点30分表示为530。注意，第二个数字表示的分钟数可能超过60，也可能是负数。

输入格式：

输入在一行中给出2个整数，分别是四位数字表示的起始时间、以及流逝的分钟数，其间以空格分隔。注意：在起始时间中，当小时为个位数时，没有前导的零，即5点30分表示为530；流逝的分钟数可能超过60，也可能是负数。

输出格式：输出四位数字表示的终止时间，当小时为个位数时，没有前导的零。题目保证起始时间和终止时间在同一天内。

输入样例 1120 110

输出样例 输出样例

**#include <stdio.h>**

**#include <math.h>**

**int main (void)**

**{ int start, min1;**

**int min2, time;**

**int end;**

**scanf("%d%d", &start, &min1);**

**min2 = start / 100 \* 60 + start % 100;**

**time = min2 + min1;**

**end = time / 60 \*100 + time % 60;**

**printf ("%d", end);**

**return 0;**

**}**

**7-3 逆序的三位数** 程序每次读入一个正3位数，然后输出按位逆序的数字。注意：当输入的数字含有结尾的0时，输出不应带有前导的0。比如输入700，输出应该是7。

输入格式：每个测试是一个3位的正整数。

输出格式：输出按位逆序的数。

输入样例： 123 输出样例：321

**#include <iostream>**

**#include <math.h>**

**using namespace std;**

**int main(){**

**int a[3] = {};**

**int x,y =0;**

**cin >> x;**

**for(int i = 0;i<3;i++)**

**{**

**a[i] = x%10;**

**x = x/10;**

**}**

**for (int i = 0;i<3;i++)**

**{**

**if (a[i]!=0)**

**{**

**y = a[i]\*pow(10,2-i)+y;**

**}**

**}**

**cout << y<<endl;**

**7-4 重要的话说三遍**

这道超级简单的题目没有任何输入。

你只需要把这句很重要的话 —— “I'm gonna WIN!”——连续输出三遍就可以了。

注意每遍占一行，除了每行的回车不能有任何多余字符。

#include <iostream>

using namespace std;

int main()

{

cout <<"I'm gonna WIN!\n" <<"I'm gonna WIN!\n" <<"I'm gonna WIN!"<< endl;

return 0;

}

**7-5 后天** 如果今天是星期三，后天就是星期五；如果今天是星期六，后天就是星期一。我们用数字1到7对应星期一到星期日。给定某一天，请你输出那天的“后天”是星期几。

输入格式：输入第一行给出一个正整数D（1 ≤ D ≤ 7），代表星期里的某一天。

输出格式：在一行中输出D天的后天是星期几。

输入样例：3 输出样例 5

#include <iostream>

using namespace std;

int main(){

int x;

cin >> x;

if(x<6){

cout << x+2 << endl;

}

else

{

cout << x-5<< endl;

}

return 0;

}

**7-6 是不是太胖了** 据说一个人的标准体重应该是其身高（单位：厘米）减去100、再乘以0.9所得到的公斤数。已知市斤的数值是公斤数值的两倍。现给定某人身高，请你计算其标准体重应该是多少？（顺便也悄悄给自己算一下吧……）

### 输入格式：输入第一行给出一个正整数H（100 < H ≤ 300），为某人身高。

### 输出格式：在一行中输出对应的标准体重，单位为市斤，保留小数点后1位。

### 输入样例：169 输出样例： 124.2

#include <iostream>

#include <math.h>

using namespace std;

int main(){

float x;

cin >> x;

float y = (x-100)\*1.8;

printf("%.1f", y);

return 0;

}

**7-7 宇宙无敌大招呼** 据说所有程序员学习的第一个程序都是在屏幕上输出一句“Hello World”，跟这个世界打个招呼。作为天梯赛中的程序员，你写的程序得高级一点，要能跟任意指定的星球打招呼。

### 输入格式：输入在第一行给出一个星球的名字S，是一个由不超过7个英文字母组成的单词，以回车结束。

### 输出格式：在一行中输出Hello S，跟输入的S星球打个招呼。

### 输入样例：Mars 输出样例：Hello Mars

#include <iostream>

using namespace std;

int main(){

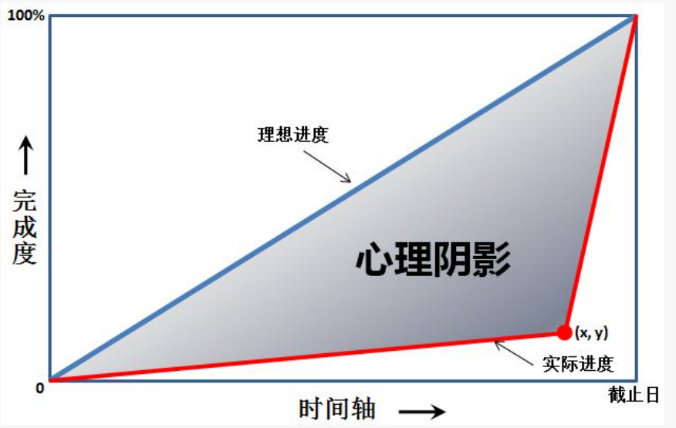
char a[20];

cin >> a;

cout << "Hello " << a <<endl;

return 0;

}

**7-8 心理阴影面积** 这是一幅心理阴影面积图。我们都以为自己可以匀速前进（图中蓝色直线），而拖延症晚期的我们往往执行的是最后时刻的疯狂赶工（图中的红色折线）。由红、蓝线围出的面积，就是我们在做作业时的心理阴影面积。

现给出红色拐点的坐标 (x,y)，要求你算出这个心理阴影面积。

### 输入格式：

输入在一行中给出 2 个不超过 100 的正整数 x 和 y，并且保证有 x>y。这里假设横、纵坐标的最大值（即截止日和最终完成度）都是 100。

### 输出格式：

在一行中输出心理阴影面积。

友情提醒：三角形的面积 = 底边长 x 高 / 2；矩形面积 = 底边长 x 高。嫑想得太复杂，这是一道 5 分考减法的题……

#include <iostream>

using namespace std;

int main(){

int x,y;

cin >> x >> y;

cout << 5000-50\*(100-x+y) << endl;

return 0;

}



**7-1 将数组中的数逆序存放**本题要求编写程序，将给定的n个整数存入数组中，将数组中的这n个数逆序存放，再按顺序输出数组中的元素。

### 输入格式:输入在第一行中给出一个正整数n（1≤n≤10）。第二行输入n个整数，用空格分开。

### 输出格式:在一行中输出这n个整数的处理结果，相邻数字中间用一个空格分开，行末不得有多余空格。

### 输入样例: 4 10 8 1 2 输出样例: 2 1 8 10

#include <iostream>

using namespace std;

int main(){

int n,i;

cin >> n;

int s[n],q[n];

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

{

cin >> s[i];

}

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

{

q[n-i-1] = s[i];

}

for(i=0;i<n-1;i++)

{

cout << q[i]<<" ";

}

cout << q[n-1];

return 0;

}

**7-2 求整数的位数及各位数字之和**对于给定的正整数N，求它的位数及其各位数字之和。

输入格式：输入在一行中给出一个不超过10​9​​的正整数N。

输出格式：在一行中输出N的位数及其各位数字之和，中间用一个空格隔开。

#include <iostream>

using namespace std;

int main(){

int N,sum = 0,n = 1;

cin >> N;

for(;N >= 10;)

{

n++;

sum = sum + (N%10);

N = N/10;

}

sum = sum +N;

cout << n <<" "<< sum<<endl;;

return 0;

}

**7-3 求整数段和** 给定两个整数A和B，输出从A到B的所有整数以及这些数的和。

### 输入格式：输入在一行中给出2个整数A和B，其中−100≤A≤B≤100，其间以空格分隔。

### 输出格式：首先顺序输出从A到B的所有整数，每5个数字占一行，每个数字占5个字符宽度，向右对齐。最后在一行中按Sum = X的格式输出全部数字的和X。

#include <iostream>

using namespace std;

int main(){

int A, B,i,Sum = 0,m;

cin >> A >>B;

int s[B-A+1];

for (i=0;i<B-A+1;i++)

{

s[i] = A+i;

Sum = Sum +s[i];

}

m = B-A+1-((B-A+1)%5);

for (i=0; i<m;)

{

printf("%5d%5d%5d%5d%5d\n",s[i],s[i+1],s[i+2],s[i+3],s[i+4]);

i = i+5;

}

if(m != B-A+1)

{

for(i=m; i<B-A+1;i++)

{

printf("%5d",s[i]);

}

cout << "\n";

}

cout << "Sum = "<< Sum<<endl;

return 0;

}

**7-4 大笨钟**微博上有个自称“大笨钟V”的家伙，每天敲钟催促码农们爱惜身体早点睡觉。不过由于笨钟自己作息也不是很规律，所以敲钟并不定时。一般敲钟的点数是根据敲钟时间而定的，如果正好在某个整点敲，那么“当”数就等于那个整点数；如果过了整点，就敲下一个整点数。另外，虽然一天有24小时，钟却是只在后半天敲1~12下。例如在23:00敲钟，就是“当当当当当当当当当当当”，而到了23:01就会是“当当当当当当当当当当当当”。在午夜00:00到中午12:00期间（端点时间包括在内），笨钟是不敲的。

下面就请你写个程序，根据当前时间替大笨钟敲钟。

### 输入格式：输入第一行按照hh:mm的格式给出当前时间。其中hh是小时，在00到23之间；mm是分钟，在00到59之间。

### 输出格式：根据当前时间替大笨钟敲钟，即在一行中输出相应数量个Dang。如果不是敲钟期，则输出Only hh:mm. Too early to Dang. 其中hh:mm是输入的时间

### #include <iostream>

### #include <stdio.h>

### using namespace std;

### int main(){

### int a,b;

### scanf("%02d:%02d",&a,&b);

### if(a<12 or (a==12 && b == 0))

### {

### printf("Only %02d:%02d. Too early to Dang.",a,b);

### }

### else

### {

### if(b != 0)

### {

### a = a+1;

### }

### for(int i=0;i<a-12;i++)

### {

### cout << "Dang";

### }

### }

### return 0;

### }

**7-5 验证“哥德巴赫猜想”** 数学领域著名的“哥德巴赫猜想”的大致意思是：任何一个大于2的偶数总能表示为两个素数之和。比如：24=5+19，其中5和19都是素数。本实验的任务是设计一个程序，验证20亿以内的偶数都可以分解成两个素数之和。

### 输入格式：输入在一行中给出一个(2, 2 000 000 000]范围内的偶数N。

### 输出格式：在一行中按照格式“N = p + q”输出N的素数分解，其中p ≤ q均为素数。又因为这样的分解不唯一（例如24还可以分解为7+17），要求必须输出所有解中p最小的解。

### #include <iostream>

### #include <math.h>

### using namespace std;

### int prime(int x){

### for(int i = 2; i <= sqrt(x); i++){

### if (x % i == 0) {

### return 0;

### }

### }

### return 1;

### }

### int main(){

### int n;

### cin >> n;

### for (int i=2;i<=(n/2);i++)

### {

### if(prime(n-i) && prime(i)){

### int a =i;

### int b= n-i;

### cout << n << " = "<< a <<" + "<<b;

### break;

### }

### }

### }

**7-6 圆形体体积计算器**本题要求实现一个常用圆形体体积的计算器。计算公式如下：

* 球体体积 V=​3​​4​​πr​3​​，其中r是球体半径。
* 圆柱体体积 V=πr​2​​h，其中r是底圆半径，h是高。
* 圆锥体体积 V=​3​​1​​πr​2​​h，其中r是底圆半径，h是高。

输入格式：

在每次计算之前，要求输出如下界面：

### Ball

### Cylinder

### Cone other-Exit

### Please enter your command:

然后从标准输入读进一个整数指令。

输出格式：

如果读入的指令是1或2或3，则执行相应的体积计算；如果是其他整数，则程序结束运行。

* 当输入为1时，在计算球体体积之前，打印Please enter the radius:，然后读入球体半径，完成计算；
* 当输入为2时，在计算圆柱体体积之前，打印Please enter the radius and the height:，然后读入底圆半径和高，完成计算；
* 当输入为3时，在计算圆锥体体积之前，打印Please enter the radius and the height:，然后读入底圆半径和高，完成计算。

计算结果在一行内输出，保留小数点后两位。

### #include<stdio.h>

### #include<math.h>

### int main(){

### int i,j,num;

### double pi=3.1415926535,r,h;

### while(1){

### printf("1-Ball\n2-Cylinder\n3-Cone\nother-Exit\nPlease enter your command:\n");

### scanf("%d",&num);

### switch(num){

### case 1:{

### printf("Please enter the radius:\n");

### scanf("%lf",&r);

### printf("%.2lf\n",4.0/3\*pi\*pow(r,3));

### break; }

### case 2:{

### printf("Please enter the radius and the height:\n");

### scanf("%lf %lf",&r,&h);

### printf("%.2lf\n",pi\*pow(r,2)\*h);

### break;

### }

### case 3:{

### printf("Please enter the radius and the height:\n");

### scanf("%lf %lf",&r,&h);

### printf("%.2lf\n",1.0/3\*pi\*pow(r,2)\*h);

### break;

### }

### default:

### return 0;

### }

### }

### return 0;

### }

**7-7 有理数均值**本题要求编写程序，计算N个有理数的平均值。

### 输入格式：输入第一行给出正整数N（≤100）；第二行中按照a1/b1 a2/b2 …的格式给出N个分数形式的有理数，其中分子和分母全是整形范围内的整数；如果是负数，则负号一定出现在最前面。

### 输出格式：在一行中按照a/b的格式输出N个有理数的平均值。注意必须是该有理数的最简分数形式，若分母为1，则只输出分子。

### #include <iostream>

### using namespace std;

### int gcd(int a , int b){

### if(b==0)

### return a;

### else

### return gcd(b,a%b);

### }

### int main(){

### int N,a1,a2,b1,b2,tmp;

### char c;

### cin >> N >>a1>>c>>b1;

### for(int i=1;i<N;++i){

### cin >> a2>>c>>b2;

### a1 = a1\*b2 +a2\*b1;

### b1 = b1\*b2;

### tmp = gcd(a1,b1);

### a1=a1/tmp,b1 = b1/tmp;

### }

### b1=b1\*N;

### tmp = gcd(a1,b1);

### a1 = a1/tmp,b1 = b1/tmp;

### if(a1%b1==0)

### cout << a1/b1;

### else

### cout << a1<<"/" <<b1;

### return 0;

### }

**7-8 组织星期信息**输入一个正整数repeat (0<repeat<10)，做repeat次下列运算：

定义一个指针数组将下面的星期信息组织起来，输入一个字符串，在表中查找，若存在，输出该字符串在表中的序号，否则输出-1。

Sunday Monday Tuesday Wednesday Thursday Friday Saturday

输入输出示例：括号内为说明，无需输入输出

### 输入样例 (repeat=3) : 3

### Tuesday

### Wednesday

### year

### 输出样例: 3

### 4

### -1

### #include <stdio.h>

### #include <string.h>

### int main()

### {

### char \*date[]={"Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"};

### int repeat,j;

### char s[100];

### scanf("%d",&repeat);

### getchar();

### for(int i=0;i<repeat;i++){

### scanf("%s",s);

### 

### for(j=0;j<7;j++){

### if(strcmp(s,date[j] )== 0){

### printf("%d\n" , j+1);

### break;

### }

### if(j=7)

### printf("-1\n");

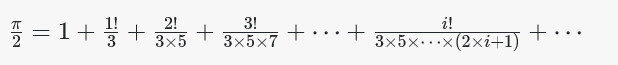
### }

### }

### return 0;

### }

### 

**7-1 近似求PI**本题要求编写程序，根据下式求π的近似值，直到最后一项小于给定精度eps。

### 输入格式：

### 输入在一行中给出精度eps，可以使用以下语句来读输入：scanf("%le", &eps);

### 输出格式：

### 在一行内，按照以下格式输出π的近似值（保留小数点后5位）：PI = 近似值

### 输入样例：1E-5 输出样例：PI = 3.14158

### #include <stdio.h>

### int main(){

### double eps, sum = 1, i, temp = 1;

### scanf("%le", &eps);

### for(i=1;temp>eps;i++){

### temp = temp\*i/(2\*i+1);

### sum += temp;

### }

### printf("PI = %.5f\n", 2\*sum);

### return 0;

### }

**7-2 整数分解为若干项之和**将一个正整数N分解成几个正整数相加，可以有多种分解方法，例如7=6+1，7=5+2，7=5+1+1，…。编程求出正整数N的所有整数分解式子。

输入格式：每个输入包含一个测试用例，即正整数N (0<<<N≤\le≤30)。

输出格式：

按递增顺序输出N的所有整数分解式子。递增顺序是指：对于两个分解序列N1=N\_1=N​1​​={n1,n2,⋯n\_1, n\_2, \cdotsn​1​​,n​2​​,⋯}和N2=N\_2=N​2​​={m1,m2,⋯m\_1, m\_2, \cdotsm​1​​,m​2​​,⋯}，若存在iii使得n1=m1,⋯,ni=min\_1=m\_1, \cdots , n\_i=m\_in​1​​=m​1​​,⋯,n​i​​=m​i​​，但是ni+1<mi+1n\_{i+1} < m\_{i+1}n​i+1​​<m​i+1​​,则N1N\_1N​1​​序列必定在N2N\_2N​2​​序列之前输出。每个式子由小到大相加，式子间用分号隔开，且每输出4个式子后换行。

输入样例：7

输出样例：

7=1+1+1+1+1+1+1;7=1+1+1+1+1+2;7=1+1+1+1+3;7=1+1+1+2+2 7=1+1+1+4;7=1+1+2+3;7=1+1+5;7=1+2+2+2

7=1+2+4;7=1+3+3;7=1+6;7=2+2+3

7=2+5;7=3+4;7=7

### #include<stdio.h>

### int count =0;

### int n,a[50];

### void dfs(int number, int n, int location, int sum){

### if(sum==n){

### printf("%d=",n);

### for(int i=0;i<location;i++)

### if(i==0)

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

### else

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

### ++count;

### if(count%4 != 0 && number != n)

### printf(";");

### if(count%4 == 0 && count!=0)

### printf("\n");

### }

### if(sum >n) return;

### for(int i=number;i<=n;++i){

### a[location]=i;

### dfs(i,n,location+1,sum+i);

### }

### }

### int main(){

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

### dfs(1,n,0,0);

### }

**7-3 特立独行的幸福**对一个十进制数的各位数字做一次平方和，称作一次迭代。如果一个十进制数能通过若干次迭代得到 1，就称该数为幸福数。1 是一个幸福数。此外，例如 19 经过 1 次迭代得到 82，2 次迭代后得到 68，3 次迭代后得到 100，最后得到 1。则 19 就是幸福数。显然，在一个幸福数迭代到 1 的过程中经过的数字都是幸福数，它们的幸福是依附于初始数字的。例如 82、68、100 的幸福是依附于 19 的。而一个**特立独行**的幸福数，是在一个有限的区间内不依附于任何其它数字的；其**独立性**就是依附于它的的幸福数的个数。如果这个数还是个素数，则其独立性加倍。例如 19 在区间[1, 100] 内就是一个特立独行的幸福数，其独立性为 2×4=8。

另一方面，如果一个大于1的数字经过数次迭代后进入了死循环，那这个数就不幸福。例如 29 迭代得到 85、89、145、42、20、4、16、37、58、89、…… 可见 89 到 58 形成了死循环，所以 29 就不幸福。

本题就要求你编写程序，列出给定区间内的所有特立独行的幸福数和它的独立性。

输入格式：

输入在第一行给出闭区间的两个端点：1<A<B≤10​4​​。

输出格式：

按递增顺序列出给定闭区间 [A,B] 内的所有特立独行的幸福数和它的独立性。每对数字占一行，数字间以 1 个空格分隔。

如果区间内没有幸福数，则在一行中输出 SAD。

输入样例 1： 10 40

输出样例 1：19 8 23 6 28 3 31 4 32 3

**注意：**样例中，10、13 也都是幸福数，但它们分别依附于其他数字（如 23、31 等等），所以不输出。其它数字虽然其实也依附于其它幸福数，但因为那些数字不在给定区间 [10, 40] 内，所以它们在给定区间内是特立独行的幸福数。

#include <bits/stdc++.h>

using namespace std;

const int maxn =10000+5;

int a[maxn], b[maxn];

bool ishappy(int t){

int k=t;

bool flag =false;

vector<int>ans(350,0);

vector<int>v;

int cnt =0;

while(2){

int n=t,a=0;

while(n!=0){ a+=(n%10)\*(n%10); n/=10; }

t=a;

if(t==1){cnt++;flag =true;break;}

else{

if(ans[t]==0) {cnt++;ans[t]=1;v.push\_back(t);}

else{flag=false;break;}

}

}

if(flag){

b[k]=cnt;

for(int i=0;i<v.size();i++) { a[v[i]]=1;}

}

return flag;

}

bool isprime(int t){

int k=sqrt(t);

if(t==2)return true;

for(int i=2;i<=k;i++){

if(t%i==0)return false;

}

return true;

}

int main(){

int n1,n2;

vector<int>happynum;

scanf("%d%d",&n1,&n2);

for(int t=n1;t<=n2;t++){

if(ishappy(t)){happynum.push\_back(t); }

}

if(happynum.size()>0){

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

int t=happynum[i];

if(a[t]==0){

printf("%d ",t);

if(isprime(t)) printf("%d\n",b[t]\*2);

else printf("%d\n",b[t]);

}

}

}else{printf("SAD");}

return 0;

}

**7-4 找出不是两个数组共有的元素** 给定两个整型数组，本题要求找出不是两者共有的元素。

### 输入格式:输入分别在两行中给出两个整型数组，每行先给出正整数N（≤20），随后是N个整数，其间以空格分隔。

### 输出格式:在一行中按照数字给出的顺序输出不是两数组共有的元素，数字间以空格分隔，但行末不得有多余的空格。题目保证至少存在一个这样的数字。同一数字不重复输出。

### 输入样例:

10 3 -5 2 8 0 3 5 -15 9 100

11 6 4 8 2 6 -5 9 0 100 8 1

输出样例:

3 5 -15 6 4 1

#include <stdio.h>

int main(){

int a[20], b[20], c[20];

int m,n,i,j,k=0;

scanf("%d", &m);

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

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

scanf("%d", &n);

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

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

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

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

if(a[i]==b[j])

break;

}

if(j>=n){

c[k]=a[i];

k++;

}

}

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

for(j=0;j<m;j++){

if(b[i]==a[j])

break;

}

if(j>=m){

c[k]=b[i];

k++;

}

}

printf("%d",c[0]);

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

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

if(c[i]==c[j])

break;

}

if(j>=i)

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

}

}

**7-5 简化的插入排序** 本题要求编写程序，将一个给定的整数插到原本有序的整数序列中，使结果序列仍然有序。

### 输入格式：输入在第一行先给出非负整数N（<10）；第二行给出N个从小到大排好顺序的整数；第三行给出一个整数X。

### 输出格式：在一行内输出将X插入后仍然从小到大有序的整数序列，每个数字后面有一个空格。

### 输入样例：

5

1 2 4 5 7

3

输出样例：

1 2 3 4 5 7

#include <stdio.h>

#include<stdlib.h>

int main(){

int n,i;

int a[10];

int b;

scanf("%d",&n);

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

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

}

scanf("%d",&b);

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

if(b>a[i])

continue;

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

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

}

a[i]=b;

break;

}

if(n==i){

a[n] = b;

}

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

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

}

return 0;

}

**7-6 冒泡法排序** 将N个整数按从小到大排序的冒泡排序法是这样工作的：从头到尾比较相邻两个元素，如果前面的元素大于其紧随的后面元素，则交换它们。通过一遍扫描，则最后一个元素必定是最大的元素。然后用同样的方法对前N−1个元素进行第二遍扫描。依此类推，最后只需处理两个元素，就完成了对N个数的排序。

本题要求对任意给定的K（<N），输出扫描完第K遍后的中间结果数列。

### 输入格式：输入在第1行中给出N和K（1≤K<N≤100），在第2行中给出N个待排序的整数，数字间以空格分隔。

### 输出格式：在一行中输出冒泡排序法扫描完第K遍后的中间结果数列，数字间以空格分隔，但末尾不得有多余空格。

### 输入样例：

6 2

2 3 5 1 6 4

输出样例：2 1 3 4 5 6

#include <stdio.h>

int main(){

int N,K,t;

scanf("%d %d",&N,&K);

int a[100];

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

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

}

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

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

if(a[j]>a[j+1]){

t = a[j+1];

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

a[j]=t;

}

}

}

for(int i=0;i<N-1;i++){

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

}

printf("%d",a[N-1]);

return 0;

}

**7-7 查验身份证** 一个合法的身份证号码由17位地区、日期编号和顺序编号加1位校验码组成。校验码的计算规则如下：

首先对前17位数字加权求和，权重分配为：{7，9，10，5，8，4，2，1，6，3，7，9，10，5，8，4，2}；然后将计算的和对11取模得到值Z；最后按照以下关系对应Z值与校验码M的值：

Z：0 1 2 3 4 5 6 7 8 9 10

M：1 0 X 9 8 7 6 5 4 3 2

现在给定一些身份证号码，请你验证校验码的有效性，并输出有问题的号码。

### 输入格式：输入第一行给出正整数N（≤100）是输入的身份证号码的个数。随后N行，每行给出1个18位身份证号码。

### 输出格式：按照输入的顺序每行输出1个有问题的身份证号码。这里并不检验前17位是否合理，只检查前17位是否全为数字且最后1位校验码计算准确。如果所有号码都正常，则输出All passed。

#include <stdio.h>

char M[]={'1','0','X','9','8','7','6','5','4','3','2'};

int Q[]={7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2};

int main(){

int n;

char num[101][19];

scanf("%d",&n);

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

scanf("%s",num[i]);

}

int sum, flag, count =0;

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

sum =0;

flag=1;

for(int j=0;j<17;j++){

if(num[i][j]>='0' && num[i][j]<='9'){

sum = sum + (num[i][j]-'0')\*Q[j];

}

else{

printf("%s\n",num[i]);

sum =0;

flag =0;

break;

}

}

if(sum>0){

sum = sum%11;

if(M[sum] != num[i][17]){

printf("%s\n",num[i]);

flag =0;

}

}

if(flag) count++;

}

if(count==n)

printf("All passed\n");

return 0;

}



**7-1 查找单价最高和最低的书籍** 编写程序，从键盘输入 n (n<10)本书的名称和定价并存入结构数组中，查找并输出其中定价最高和最低的书的名称和定价。

输出格式语句：

printf("highest price: %.1f, %s\n", );

printf("lowest price: %.1f, %s\n",);

输入输出示例：括号内为说明，无需输入输出

### 输入样例:

3 (n=3)

Programming in C

21.5

Programming in VB

18.5

Programming in Delphi

25

输出样例:

highest price: 25.0, Programming in Delphi

lowest price: 18.5, Programming in VB

#include <bits/stdc++.h>

using namespace std;

struct book{

char s[100];

double price;

}b[15];

int main()

{

int n;

scanf("%d",&n);

double min = 9999, max=0;

int maxi,mini;

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

{

getchar();

cin.getline(b[i].s,31);

cin>>b[i].price;

if(b[i].price>max){

maxi = i;

max = b[i].price;

}

if(b[i].price<min){

mini = i;

min = b[i].price;

}

}

printf("highest price: %.1f, %s\n",max,b[maxi].s);

printf("lowest price: %.1f, %s\n",min,b[mini].s);

return 0;

}

**7-2 宿舍谁最高？**学校选拔篮球队员，每间宿舍最多有4个人。现给出宿舍列表，请找出每个宿舍最高的同学。定义一个学生类Student,有身高height，体重weight等。

### 输入格式:首先输入一个整型数n （1<=n<=1000000），表示n位同学。 紧跟着n行输入,每一行格式为：宿舍号，name,height,weight。 宿舍号的区间为[0,999999]， name 由字母组成，长度小于16，height，weight为正整数。

### 输出格式:按宿舍号从小到大排序，输出每间宿舍身高最高的同学信息。题目保证每间宿舍只有一位身高最高的同学。

### 输入样例:

7

000000 Tom 175 120

000001 Jack 180 130

000001 Hale 160 140

000000 Marry 160 120

000000 Jerry 165 110

000003 ETAF 183 145

000001 Mickey 170 115

输出样例:

000000 Tom 175 120

000001 Jack 180 130

000003 ETAF 183 145

#include <iostream>

#include <string>

#include<algorithm>

#include <cstdio>

#define maxn 100

using namespace std;

class Student{

public:

Student(int room, string name, int height, int weight){

this->room = room;

this->name = name;

this->height = height;

this->weight = weight;

}

Student (){}

int room;

string name;

int height;

int weight;

};

bool cmp(Student a, Student b){

if(a.room != b.room){

return a.room < b.room;

}

else{

return a.height < b.height;

}

}

Student s[maxn];

int main(){

int n;

cin >> n;

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

int r,h,w;

string sname;

cin >>r>>sname>>h>>w;

s[i] = Student(r,sname,h,w);

}

sort(s,s+n,cmp);

int i = 0;

while(i<n){

int j = i+1;

while(s[i].room == s[j].room){

j++;

}

i = j;

printf("%06d ", s[j-1].room);

cout << s[j-1].name << " "<<s[j-1].height<<" "<<s[j-1].weight<<endl;

}

return 0;

}

**7-3 鸿鸿哥的苹果树** 鸿鸿哥家的院子里有一棵桃子树，每到秋天树上就会结出10个桃子。桃子成熟的时候，鸿鸿哥就会跑去摘桃子。鸿鸿哥有个30厘米高的板凳，当他不能直接用手摘到桃子的时候，就会踩到板凳上再试试。 现在已知10个桃子到地面的高度，以及鸿鸿哥把手伸直的时候能够达到的最大高度，请帮鸿鸿哥算一下他能够摘到的桃子的数目。假设他碰到桃子，桃子就会掉下来。

### 输入格式:

第一行中给出10个桃子到地面的高度(cm)。

第二行中输入鸿鸿哥把手伸直能够达到的最大高度(cm)。

### 输出格式:

输出鸿鸿哥能够摘到的桃子数目n。

### 输入样例:在这里给出一组输入。例如：26 95 43 77 49 31 87 19 35 65

40

### 输出样例: 在这里给出相应的输出。例如：7

#include <iostream>

using namespace std;

class Peach{

public:

int length[10];

};

int main(){

Peach p;

for(int i=0;i<10;i++){

cin>>p.length[i];

}

const int stool = 30;

int height;

cin>>height;

int H = height + stool;

int count = 0;

for(int i=0;i<10;i++){

if(H>=p.length[i])

count++;

}

cout <<count<<endl;

return 0;

}

**7-4 返回日期的星期信息** 定义一个日期类CData，其有数据成员year,month,day,函数成员有构造函数，析构函数，复制构造函数，显示日信息函数display（年月日星期）. 函数Getweek是返回改日期是英文单词表示的星期几。从键盘读入一个日期，先显示21世纪的第一天日期的信息（已知2000年1月1号是星期六），然后再建立一个新日期的信息。 主函数如下无需重写： //主函数 int main() {  
int y,m,d; cin>>y>>m>>d; CDate d0; d0.display(); cout<<endl; CDate d1(y,m,d); d1.display(); }

### 输入格式:

输入一个日期（2000-1-1起的一个日期，保证输入日期合法有效） 之间以空格隔开

### 输出格式: 先输出21世纪第一天的日期信息 而后输出输入指定日期的信息

### 输入样例: 在这里给出一组输入。例如：2000 1 31

### 输出样例:在这里给出相应的输出。例如

The first day of the 21st century:

2000-1-1:Saturday

Create a new date :

2000-1-31:Monday

#include <iostream>

using namespace std;

class CDate {

private:

int year, month, day;

public:

CDate() {

year = 2000;

month = 1;

day = 1;

cout << "The first day of the 21st century:" << endl;

}

CDate(int y, int m, int d) {

cout << "Create a new date :" << endl;

year = y;

month = m;

day = d;

}

void display() {

cout <<year<<"-"<<month<<"-"<<day<< ":";

int num = 0;

int a=0,b=0,c=0,d=0;

a=year-2000;

if(a%4==0){

b=(a-a%4)/4\*366;

c=a-(a-a%4)/4;

}

else{

b=((a-a%4)/4+1)\*366;

c=a-(a-a%4)/4-1;

}

d=c\*365;

num=b+d;

for (int i = 1; i < month; ++i) {

if (i == 1 || i == 3 || i == 5 || i == 7 || i == 8 || i == 10 || i == 12) {

num += 31;

} else if (i == 4 || i == 6 || i == 9 || i == 11) {

num += 30;

} else if ((year - 2000) % 4) {

num += 28;

} else {

num += 29;

}

}

num +=(day-1);

switch (num % 7) {

case 0:

cout << "Saturday";

break;

case 1:

cout << "Sunday";

break;

case 2:

cout << "Monday";

break;

case 3:

cout << "Tuesday" ;

break;

case 4:cout << "Wednesday";

break;

case 5:

cout << "Thursday";

break;

case 6:

cout << "Friday";

break;

}

}

};

int main() {

int y, m, d;

cin >> y >> m >> d;

CDate d0;

d0.display();

cout<<endl;

CDate d1(y, m, d);

d1.display();

}

**7-5 日程安排（多重继承+重载）**

已有一个日期类Date，包括三个protected成员数据

int year;

int month;

int day;

另有一个时间类Time，包括三个protected成员数据

int hour;

int minute;

int second;

现需根据输入的日程的日期时间，安排前后顺序，为此以Date类和Time类为基类，建立一个日程类Schedule，包括以下新增成员：

int ID；//日程的ID

bool operator < (const Schedule & s2);//判断当前日程时间是否早于s2

生成以上类，并编写主函数，根据输入的各项日程信息，建立日程对象，找出需要最早安排的日程，并输出该日程对象的信息。

输入格式： 测试输入包含若干日程，每个日程占一行（日程编号ID 日程日期（\*\*//）日程时间（::））。当读入0时输入结束，相应的结果不要输出。

输入样例：

1 2014/06/27 08:00:01

2 2014/06/28 08:00:01

0

输出样例：

The urgent schedule is No.1: 2014/6/27 8:0:1

#include<iostream>

#include <stdio.h>

using namespace std;

class Data {

protected:

int year;

int month;

int day;

public:

Data() {}

Data(int y, int mo, int d) {

year = y;

month = mo;

day = d;

}

void printdata() {

cout << " " << year << "/" << month << "/" << day;

}

};

class Time {

protected:

int hour;

int minute;

int second;

public:

Time() {}

Time(int h, int mi, int s) {

hour = h;

minute = mi;

second = s;

}

void printtime() {

cout << " " << hour << ":" << minute << ":" << second;

}

};

class Schedule :public Data, public Time {

int ID;

public:

Schedule() {}

Schedule(int y, int mo, int d, int h, int mi, int s, int id) :Data(y, mo, d), Time(h, mi, s) {

ID = id;

}

void printschedule() {

cout << "No." << ID << ":";

Data::printdata();

Time::printtime();

}

bool operator<(const Schedule& s2);

};

bool Schedule::operator<(const Schedule& s2) {

if (year < s2.year) {

return 1;

}

else if (year > s2.year){

return 0;

}

else {

if (month < s2.month){

return 1;

}

else if (month > s2.month) {

return 0;

}

else {

if (day < s2.day){

return 1;

}

else if (day > s2.day) {

return 0;

}

else {

if (hour < s2.hour){

return 1;

}

else if (hour > s2.hour) {

return 0;

}

else {

if (minute < s2.minute) {

return 1;

}

else if (minute > s2.minute) {

return 0;

}

else {

if (second < s2.second){

return 1;

}

else{

return 0;

}

}

}

}

}

}

}

int main() {

int id, y, mon, d, h, min, s, i = 0;

Schedule s2(9999, 9999, 9999, 999, 999, 99,0);

while (1)

{

cin >> id;

if (id == 0)

break;

i++;

scanf("%d/%d/%d", &y, &mon, &d);

scanf("%d:%d:%d", &h, &min, &s);

Schedule s1(y, mon, d, h, min, s,id);

if (s1 < s2)

s2 = s1;

}

if (i != 0)

{

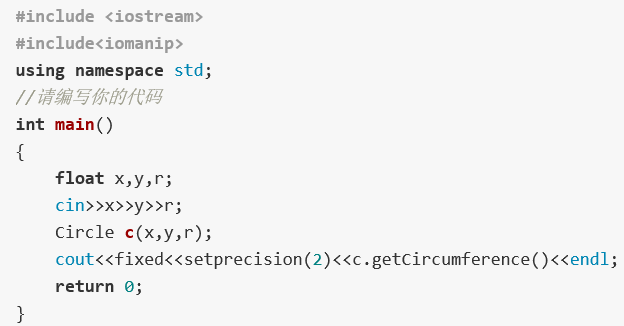
cout << "The urgent schedule is "; s2.printschedule();

}

return 0;

}

**7-6 定义基类Point和派生类Circle，求圆的周长.** 定义基类Point（点）和派生类Circle（圆），求圆的周长。Point类有两个私有的数据成员float x,y;Circle类新增一个私有的数据成员半径float r和一个公有的求周长的函数getCircumference();主函数已经给出，请编写Point和Circle类。



### 输入格式: 输入圆心和半径，x y r中间用空格分隔。

### 输出格式:输出圆的周长，小数点后保留2位有效数字。

输入样例: 1 2 3

输出样例:

Point constructor called

Circle constructor called

18.84

Circle destructor called

Point destructor called

#include <iostream>

#include<iomanip>

#define PI 3.14

using namespace std;

class Point{

protected:

float X, Y;

public:

Point() {}

Point(float x, float y) :X(x), Y(y) {

cout << "Point constructor called"<<endl;

}

~Point(){

cout <<"Point destructor called"<<endl;

}

};

class Circle :public Point{

float R;

public:

float c;

Circle() {}

Circle(float x, float y, float r) :Point(x,y),R(r) {

cout << "Circle constructor called"<<endl;

}

~Circle(){

cout <<"Circle destructor called"<<endl;

}

float getCircumference();

};

float Circle::getCircumference(){

c = PI \* R \* 2;

return c;

}

int main()

{

float x,y,r;

cin>>x>>y>>r;

Circle c(x,y,r);

cout<<fixed<<setprecision(2)<<c.getCircumference()<<endl;

return 0;

}

**7-7 用虚函数计算各种图形的面积**定义抽象基类Shape，由它派生出五个派生类：Circle（圆形）、Square（正方形）、Rectangle（ 长方形）、Trapezoid （梯形）和Triangle （三角形），用虚函数分别计算各种图形的面积，输出它们的面积和。要求用基类指针数组，每一个数组元素指向一个派生类的对象。PI=3.14159f，单精度浮点数计算。

### 输入格式:输入在一行中，给出9个大于0的数，用空格分隔，分别代表圆的半径，正方形的边长，矩形的宽和高，梯形的上底、下底和高，三角形的底和高。

### 输出格式:输出所有图形的面积和，小数点后保留3位有效数字。

### 输入样例: 12.6 3.5 4.5 8.4 2.0 4.5 3.2 4.5 8.4

### 输出样例: 578.109

#include <iostream>

#include<iomanip>

using namespace std;

#define PI 3.14159f

class Shape

{public:

virtual double area() const =0;

};

class Circle:public Shape

{public:

Circle(double r):radius(r){}

virtual double area() const {return PI\*radius\*radius;};

protected:

double radius;

};

class Square:public Shape

{public:

Square(double s):side(s){}

virtual double area() const {return side\*side;}

protected:

double side;

};

class Rectangle:public Shape

{public:

Rectangle(double w,double h):width(w),height(h){}

virtual double area() const {return width\*height;}

protected:

double width,height;

};

class Trapezoid:public Shape

{public:

Trapezoid(double t,double b,double h):top(t),bottom(b),height(h){}

virtual double area() const {return 0.5\*(top+bottom)\*height;}

protected:

double top,bottom,height;

};

class Triangle:public Shape

{public:

Triangle(double w,double h):width(w),height(h){}

virtual double area() const {return 0.5\*width\*height;}

protected:

double width,height;

};

int main()

{

float a,b,c,d,e,f,g,h,i;

cin>>a>>b>>c>>d>>e>>f>>g>>h>>i;

Circle circle(a);

Square square(b);

Rectangle rectangle(c,d);

Trapezoid trapezoid(e,f,g);

Triangle triangle(h,i);

Shape \*pt[5]={&circle,&square,&rectangle,&trapezoid,&triangle};

float areas=0.0;

for(int i=0;i<5;i++){

areas=areas+pt[i]->area();

}

cout<<fixed<<setprecision(3)<<areas<<endl;

return 0;

}