蓝桥

LAN QIAO

蓝桥杯大赛

青少年创意编程

【真题一本通】

目 录

比赛中的注意事项（C/C++）

蓝桥杯 C++项目题库

蓝桥杯官网练习系统基础题目（选拔赛省赛必看）

蓝桥杯白皮书 C++编程题整理

蓝桥杯官网练习系统 VIP 题目题解

（省赛高级组或国赛初级中级组必看）

蓝桥杯大赛青少组 C++组赛前集训包

2 比赛中的注意事项（C/C++）

2.1 关于时间安排

比赛时间 2 个小时，注意以提交题目的系统时间为准，做完一题交一题，避免最后因为时间不够没交完题。在 2 个小时中同一道题可以提交多次，每次提交都会覆盖上一次的代码。

2.2 关于填空题

选拔赛 STEMA 考试内容分为两部分：第一部分为科技素养及逻辑思维，45 分钟；第二部分 为程序设计，75 分钟。选择题目答案唯一，正确得全部计分点，空白不得分， 错误扣 1 分。

解决填空题一定要用最快的方法，能用工具解决的题绝不写代码，比如计算器，Excel 表格。

每一道填空题看一遍如果有思路就写，没思路直接跳过，科学素养题目答错扣一分，不答不扣分。

填空题尽量控制在 30 分钟以后，把自己该拿的分拿到。

2.3 关于编程大题

每一道编程大题程序结束，一定要写 return 0; 不写整道题都没分！推荐一个好的编程习惯：先写出主程序的框架，再去补中间的代码：

```
#include <iostream>
using namespace std;
int main()
{
    //代码区
      return 0;
}
```

编程大题中的头文件

万能头文件写法：#include <bits/stdc++.h>

建议大家在平时刷题和比赛过程中还是把所有头文件都写出来，如果实在记不住头文件，再使用万能头文件。因为使用万能头文件编译时间可能略长，造成程序超时。

编程大题中的输入输出

① 由于是 C 和 C++混合编程，所以可以使用 C 语言中的 scanf 和 printf 来进行输入输出，也可以使用 C++中的 cin 和 cout 进行输入输出。

但是强烈推荐使用 scanf 和 printf 来输入输出。因为使用 cin 和 cout 存在一个缓冲

区的刷新，时间较慢，可能会造成程序超时。

② 蓝桥杯比赛中多数是单输入和单输出，也就是只要求输入一个数并输出一个结果，而平时做题一般是多输入和多输出，也就是多组数据对应多个输出结果。注意：在多输入和多输出的题目中，在每一次循环后临时变量是否需要清零，是否需要清空标记等一系列问题。

③ 关于数据范围：大题的题目中会给出相应的数据范围，根据题目的数据范围来选择数据型是 int 还是 long long 还是其他类型 float 和 double。

④ 注意输入输出的格式，看清输出是否有空格或者是行末空格，如果有的话对于行末要进行特殊处理（加个 if 语句就行）。

⑤ 建议一些变量、数组、标记等定义在全局，这样就避免了初始化，因为放在全局的变量，系统自动初始化成 0。

关于模板题

蓝桥杯常考模板题，对于一些常见模板，我们要做到烂熟于心。比如 DFS 和 BFS 的模板、二分模板、背包问题模板等等，这些模板在学习时要理解算法思想，然后通过不断练习来背过模板，在比赛中看到类似的题能加快速度。

蓝桥杯重点考察的就是程序设计能力。选拔赛要求熟悉最大公约数、日期计算、数的进制字符串转化等基础算法。蓝桥杯竞赛采用题库抽题组题方式，题目重现几率比较大，历届真题都要了如指掌，还要能举一反三。

关于 OJ 在线评测训练

在 C 语言网，https://www.dotcpp.com 训练-题库 可以很方便的找到近几年蓝桥杯比赛真题，只要在题目中用“蓝桥杯 2021”搜索下，就可以看到当年真题，可以在线评测。在洛谷网站 https://www.luogu.com.cn/ 也有类似题目，但比较少，优点在于有大牛出题解，可以观摩。

2.4 关于样例

样例是一个比较坑的东西，有的题样例过了，但照样是零分。比赛时如果时间充裕注意多测几组特殊的样例，比如 0 这样的一些特殊数据或者是一些小数据。

2.5 关于骗分

蓝桥杯比赛中的大题并不是一个样例不通过就没有分，对于一些实在做不出来的大题可以通过暴力枚举等方式来获得部分分数。

2.6 最后总结

蓝桥杯在大学生圈子里的影响力非常大，开始下沉到中小学生市场

倪光南院士站台，各大厂背景，蓝桥杯的证书，有敲门砖作用

C++项目刚起步，预计会大概率参考 NOIP 和 CSP-J/S 的题目

3 蓝桥杯 C++项目题库

3.1 ***选拔赛难度模拟题

3.1.1 圆的面积和周长

输入圆的半径 r(可能是小数)，计算圆的面积和周长。

圆的面积=PI*r*r，圆的周长=2*PI*r

输出结果保留两位小数,PI=3.14

输入

1

输出

3.14 6.28

样例输入

1.9

样例输出

11.34 11.93

```
#include<iostream>
#include<cstdio>
using namespace std;
int main()
{
    double r,pi=3.14;
    cin>>r;
    printf("%0.2f %0.2f",pi*r*r,2*pi*r);
    return 0;
}
```

3.1.2 四方定理

“四方定理”是数论中著名的一个定理，指所有自然数至多只要用四个数的平方和就可以表示。

输入 4 个自然数，输出这四个自然数的平方和

输入

7 7 7 8

输出

211

样例输入

3 4 5 7

样例输出

99

```
#include<iostream>
using namespace std;
int main()
{
  int a,b,c,d;
  cin>>a>>b>>c>>d;
  cout<<a*a+b*b+c*c+d*d;
 return 0;
}
```

3.1.3 水仙花数

水仙花数（Narcissistic number）也被称为超完全数字不变数（pluperfect digital invariant, PPDI）、自恋数、自幂数、阿姆斯壮数或阿姆斯特朗数（Armstrong number），水仙花数是指一个 3 位数，它的每个位上的数字的 3 次幂之和等于它本身（例如：1^3 + 5^3+ 3^3 = 153）。

输入一个 3 位数判断是否是水仙花数

是则输出 yes,否则输出 no

输入 153

输出 yes

样例输入 219

样例输出 no

```
#include<iostream>
using namespace std;
int main()
{
    int a,b,c,n;
    cin>>n;
    a=n/100;
    b=n/10%10;
    c=n%10;
    if(a*a*a+b*b*b+c*c*c==n)
        cout<<"yes"<<endl;
```

```
    else
        cout<<"no";
    return 0;
}
```

3.1.4 大小写转换

第一个整数 n,代表接下来有 n 个字符数据，如果是大写字符,需要把它转换成小写字符并输出

如果是小写字符,需要把它转换成大写字符并输出，如果是数字,不做任何处理,直接输出

输入

3 a F 9

输出

A

f

9

```
#include<iostream>
using namespace std;
int main()
{
    int n,i;
    char a;
    cin>>n;
    for(i=1;i<=n;i++)
    {
        cin>>a;  //大写 A 65  小写 a 97   ascii 码相差 32
        if(a>='A'&&a<='Z' ) cout<<char(a+32)<<endl;
       if(a>='a'&&a<='z' ) cout<<char(a-32)<<endl;
       if(a>='0'&&a<='9' ) cout<<a<<endl;
    }
    return 0;
}
```

3.1.5 国庆节是星期几

1949 年的国庆节（10 月 1 日）是星期六,输入一个大于 1949 年的年份 n 输出 n 年的 10 月 1 日是星期几

星期一 输出 1

星期二 输出 2

...

星期日 输出 0

输入

1950

样例输入

2019

样例输出

2

提示

计算 1949 年以后每年的天数(闰年 366 天,平年 365 天)

把天数求和然后加上 6(1949 年的国庆节是星期 6)

用这个数字对 7 求余数,就是结果

```
//整年计算，计算有多少天
#include<iostream>
using namespace std;
int main() {
    int n, i, day = 0;
    cin >> n;
    for (i = 1949; i <= n; i++) {
        //平年闰年判断
        if ((i % 4 == 0 && i % 100 != 0) || i % 400 == 0)
            day = day + 366;
        else
            day = day + 365;
    }
    cout << (day-1 + 6) % 7; //减去加重 1 天，把天数求和然后加上 6
    return 0;
}
```

3.1.6 回文素数

输入一个五位数,判断是否是回文素数，回文素数的条件：

1.是一个素数

2.第一位和第五位相等,第二位和第四位相等

输入

30203

输出

yes

样例输入

10101

样例输出

no

```
#include<iostream>
#include<cmath> //sqrt()函数用到此头文件
using  namespace  std;
int prime(int n)
{
   int m=sqrt(n);
   for(int i=2;i<=m;i++)
   {
      if(n%i==0) return 0;
   }
   return 1;
}
int  main()
{
      int   n,a,b,c,d;
      cin>>n;
      a=n%10;
      b=n/10%10;
      c=n/1000%10;
      d=n/10000%10;
      if(a==d&&b==c&&prime(n)==1)
          cout<<"yes";
          else
```

```
        cout<<"no";
    return  0;
}
```

3.1.7 数据加密

某个公司采用公用电话传递数据，数据是四位的整数，在传递过程中是加密的，加密规则如下：每位数字都加上 5,然后除以 10 的余数代替该数字，再将第一位和第四位交换，第二位和第三位交换。

输入一个四位数字，输出加密后的四位数字

输入

9876

输出

1234

样例输入

1234

样例输出

9876

```
#include<iostream>
#include<cmath>
using  namespace  std;
int  main()
{
    int n,Qw, Bw ,Sw ,Gw;
    cin>>n;
    Gw=n%10;
    Sw=n/10%10;
    Bw=n/100%10;
    Qw=n/1000%10;
    Gw=(Gw+5)%10;
    Sw=(Sw+5)%10;
    Bw=(Bw+5)%10;
    Qw=(Qw+5)%10;
    cout<< Gw << Sw <<Bw <<Qw <<endl;
   return  0;
}
```

3.1.8 蜘蛛和蜻蜓（枚举）

蜘蛛有 8 条腿，蜻蜓有 6 条腿和 2 对翅，蝉有 6 条腿和 1 对翅。三种虫子共 18 只，共有 118 条腿和 20 对翅。问每种虫子各几只？

输出蜘蛛数量,蜻蜓数量,蝉的数量　　多组数据,需换行

```
#include<iostream>
using namespace std;
int main()
{
    int zz,qt,ch;
    int ZZ,QT,CH;
    for(zz=0;zz<=18;zz++)
    {
       for(qt=0;qt<=18;qt++)
       {
          ch=18-zz-qt;
          if((zz*8+qt*6+ch*6==118) && (zz*0+qt*2+ch*1==20))
            {
              cout<<zz <<" "<<qt<<" "<<ch<<endl;
              ZZ=zz; QT=qt; CH = ch;
          }
       }
    }
    cout<<ZZ <<" "<<QT<<" "<<CH<<endl;
    cout<<ZZ*8+QT*6+CH*6<<endl;
    //cout<<zz*8+qt*6+ch*6<<endl;  为什么输出不对
    //cout<<zz*0+qt*2+ch*1<<endl;
    return 0;
}
```

3.1.9 求三位数

有一个三位数，个位数字比百位数字大，而百位数字又比十位数字大，并且各位数字之和等于各位数字相乘之积，求此三位数

方法一：

```
#include<iostream>
using  namespace  std;
int  main()
{
```

```
    int n,Gw,Bw,Sw;
    for(int n=100;n<=999;n++)
    {
        Gw=n%10;
        Sw=n/10%10;
        Bw=n/100%10;
        if(Gw+Bw+Sw==Gw*Bw*Sw&&Gw>Bw&&Bw>Sw)
            cout<<n<<endl;
        }
        return  0;
}
```

方法二：

```
#include<iostream>
using namespace std;
int main() {
    int  a,b,c,i;
    for(c=1; c<=9; c++)
        for(b=1; b<=9; b++)
            for(a=1; a<=9; a++) {
                if(a>c && c>b && a+b+c==a*b*c)
                    cout<<c*100+b*10+a<<endl;
            }
    return 0;
  }
```

3.1.10鸡兔又同笼

有一群鸡和一群兔，它们的只数相同，它们的脚数都是三位数，且这两个三位数的数字分别是 0，1，2，3，4，5。问鸡和兔的只数各是多少?它们的脚数各是多少?

输出

第一行输出鸡的数量,兔的数量

第二行输出鸡的腿数,兔的腿数

答案输出： 76 76

152 304

参考代码：枚举法

```
#include <iostream>
using namespace std;
int main() {
```

```
    int a,b,c,d,e,f;
    int j,t;
    for(a=1; a<=5; a++)
        for(b=0; b<=5; b++)
            for(c=0; c<=5; c++)  //外三层枚举鸡的脚数
                for(d=1; d<=5; d++)
                    for(e=0; e<=5; e++)
                        for(f=0; f<=5; f++) 内三层枚举兔的脚数
                        {
                            j=a*100+b*10+c;//鸡的脚数
                            t=d*100+e*10+f;//兔的脚数;
                            if(j%2==0&&t%4==0&&j/2==t/4)

    if(a!=b&&a!=c&&a!=d&&a!=e&&a!=f&&b!=c&&b!=d&&b!=e&&b!=f&&c!=d&&c!=e&&c!
=f&&d!=e&&d!=f&&e!=f)
                        { //if 嵌套
                                    cout<<j/2<<" "<<t/4<<endl;
                                    cout<<j<<" "<<t<<endl;
                                }
                        }
    return 0;
}
```

3.1.11数列特征

输入:
第一行为整数 n，表示数的个数。
第二行有 n 个整数，为给定的 n 个数，每个数的绝对值都小于 10000。
输出:
第一行输出这些数中的最大值
第二行输出这些数中的最小值
第三行输出这些数的和
第四行输出平均值(保留两位小数)
第五行输出与平均值的差在 2(包含 2)以内的数据的数量
输入
3
1 2 3
输出
3
1
6
2.00
3
样例输入
5
9 7 8 3 4
样例输出
9
3
31
6.20
2
提示
printf("%.2f\n",avg);保留两位小数请使用 printf 输出

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

```
int main()
{
    int i,n,count=0;
    long long sum=0;
    double avg;
    cin>>n;
    int a[n];
     for(i=0;i<n;i++)
    {
        cin>>a[i];
        sum=sum+a[i];
    }
    avg=sum*1.0/n;
    sort(a+1,a+n+1);
    for(i=0;i<n;i++)
    {
        if(a[i]-avg<=2) count++;
    }
    cout<<a[n-1]<<endl;
    cout<<a[0]<<endl;
    cout<<sum<<endl;
    printf("%0.2f\n",avg);
    cout<<count<<endl;
    return 0;
}
```

3.1.12插队问题

有 n 个人（每个人有一个唯一的编号，用 1~n 之间的整数表示）在一个水龙头前排队准备接水，现在第 n 个人有特殊情况，经过协商，大家允许他插队到第 x 个位置。输出第 n 个人插队后的排队情况。

输入格式

第一行 1 个正整数 n，表示有 n 个人，2<n≤100。

第二行包含 n 个正整数，之间用一个空格隔开，表示排在队伍中的第 1~ 第 n 个人的编号。

第三行包含 1 个正整数 x，表示第 n 个人插队的位置，1≤x<n。

输出格式

一行包含 n 个正整数，之间用一个空格隔开，表示第 n 个人插队后的排队情况。

输入样例

7

7 2 3 4 5 6 1

3

输出样例

7 2 1 3 4 5 6

解析：这样的题目看文字比较模糊，直接看输入输出样例。

用数组存储数字，0 的位置空出。

插队后 x-1 之前位置数字不变，x 到 n-1 位置数字均向后窜一位，用 n 位置数字覆盖 x 位置数字。

方法一、更新数组 模拟算法

```
#include<iostream>
using namespace std;
int main() {
    int a[105];
    int x,n,i,temp;
    cin>>n;
    for(i=1; i<=n; i++) {
        cin>>a[i];    //存储
    }
    temp=a[n]; //保留现场
    cin>>x;
    for(i=n; i>=x; i--) {
        a[i]=a[i-1];    //从后面开始遍历挪位置
    }
    a[x]=temp;//n 覆盖 x
    for(i=1; i<=n; i++) {
        cout<<a[i]<<" ";    //遍历输出
    }
    return 0;
}
```

方法二、拼接输出 数据量大时效率高

```
#include<iostream>
using namespace std;
int main() {
    int a[105];
    int x,n,i;
    cin>>n;
    for(i=1; i<=n; i++) {
        cin>>a[i];    //存储
    }
    cin>>x;
    for(i=1; i<x; i++)
        cout<<a[i]<<" "; //遍历输出插队位置前数据
    cout<<a[n]<<" "; //输出插队的数据
    for(i=x; i<n; i++)
        cout<<a[i]<<" "; //遍历输出插队位置后不包含 n 的数据
    return 0;
}
```

3.1.13数字和

输入一个整数 n,求各个位上的数字和,最长 200 位

输入

12345678998765432112345678998765432 1

输出

180

样例输入

456764756456435354790970910921987217532974094430939834 32

样例输出

268

注意题目说明 “ 最长 200 位”

先上一个最长 19 位 long long 数据类型能存下的解法，看一下解题思路。

```
#include <iostream>
using namespace std;
int main() {
    long long n;
```

```
    int sum = 0;
    cin>>n;
    while(n!=0) {
        sum += n%10;
        n = n/10;
    }
    cout<<sum<<endl;
    return 0;
}
```

这种用 while 取数字的方法一定要掌握

```
#include <iostream>
using namespace std;
int main() {
    string s; //用字符串存储超长的数字
    int len,sum=0;
    getline(cin, s);//接受一个字符串，可以接受空格并输出
    len=s.size();//计算 string 中的元素个数用 size()
    for(int i=0; i<len; i++)
        sum+=s[i]-'0';
    //s 是字符数组，要它变成数字，需要减去一个字符 0，完成隐式转换
    cout<<sum<<endl;
    return 0;
}
```

3.1.14字符串比较（源自蓝桥杯 VIP 题库）

给定两个仅由大写字母或小写字母组成的字符串(长度介于 1 到 10 之间)，它们之间的关系是以下 4 种情况之一：

1：两个字符串长度不等。比如 Beijing 和 Hebei

2：两个字符串不仅长度相等，而且相应位置上的字符完全一致(区分大小写)，比如 Beijing 和 Beijing

3：两个字符串长度相等，相应位置上的字符仅在不区分大小写的前提下才能达到完全一致（也就是说，它并不满足情况 2）。比如 beijing 和 BEIjing

4：两个字符串长度相等，但是即使是不区分大小写也不能使这两个字符串一致。比如 Beijing 和 Nanjing

编程判断输入的两个字符串之间的关系属于这四类中的哪一类，给出所属的类的编号
输入:
BEIjing
beiJing
输出:
3
输入
Beijing
Hebei
输出
1
样例输入
Beijing
Beijing
样例输出
2

3.1.15奖券数目

有些人很迷信数字，比如带“4”的数字，认为和“死”谐音，就觉得不吉利。虽然这些说法纯属无稽之谈，但有时还要迎合大众的需求。某抽奖活动的奖券号码是5 位数（10000-99999），要求其中不要出现带“4”的号码，主办单位请你计算一下，如果任何两张奖券不重号，最多可发出奖券多少张。
请提交该数字（一个整数），不要写任何多余的内容或说明性文字。

```
#include<iostream>
using namespace std;
int main()
{
    int ans = 0;
    for (int i = 10000; i <= 99999; i++)
    {
        int num = i;
        int a = num % 10;     num=num/10;
        int b = num % 10;    num=num/10;
        int c = num % 10; num=num/10;
        int d = num % 10; num=num/10;
```

```
        int e = num % 10;
        if(a==4||b==4||c==4||d==4||e==4)
        {
        }
        else
        ans++;
    }
    cout <<ans;
    return 0;
}
```

方法 2

```
#include<iostream>
using namespace std;
int main()
{
    int ans = 0;
    int cnt = 90000;
    for (int i = 10000; i <= 99999; i++)
    {
        int num = i;
        int a = num % 10;    num=num/10;
        int b = num % 10;   num=num/10;
        int c = num % 10;num=num/10;
        int d = num % 10;num=num/10;
        int e = num % 10;
        if(a==4||b==4||c==4||d==4||e==4)
        {
            cnt--;
        }
        else
        ans++;
    }
    cout <<ans<<endl;
    cout <<cnt<<endl;
    return 0; }
```

3.1.16纪念日

2020 年 7 月 1 日是中国共产党成立 99 周年纪念日。中国共产党成立于 1921 年 7 月 23 日。请问从 1921 年 7 月 23 日中午 12 时到 2020 年 7 月 1 日中午 12 时一共包含多少分钟？

本题的结果为一个整数，在提交答案时只填写这个整数，填写多余的内容将无法得分。

```
#include<iostream>
using namespace std;
int days(int n)
{
    if(n%400==0||n%4==0&&n%100!=0)
    return 366;
    else
    return 365;
}
int main()
{
    int sum=0;
    for(int i=1922; i<=2020; i++)
    {
        sum=sum+days(i);
    }
    sum=sum-22;
    cout<<sum*60*24;
    return 0;
}
```

3.2 ****第十一届蓝桥杯青少组 C++选拔赛中级组

3.2.1 参赛组别推荐

2121 年 8 月，为了让选手们获得更好的参赛体验，蓝桥杯大赛青少年创意编程 C++组细分为初级组和高级组，小蓝第一次报名参 C++组的比赛，不知道自己该选择哪一组参赛，你能给他一些建认吗？蓝桥杯规定 8 到 12 岁的选手可以参加初级组：13 到 18 岁的选手可以参加高级组的此赛：如果小于 8 岁，告诉他：“他还太小”：如果大于 18 岁，告诉他：“他太大了”。

输入：一个整数 n（1<n<100），代表小蓝的年龄。

输出：

请给出参赛建议：

小于 8 岁，输出信息为："You are too young！"

8 到 12 岁，输出信息为："Welcome to Junior Class！"

13 到 18 岁，输出信息为："Welcome to Senior Class！"

19 岁及以上，输出信息为："You are too old！"

样例輸入：

9

样例输出：

Welcome to Junior Class！

3.2.2 还差几天到 2022 年？

时光飞逝，转眼间就到了 2021 年的最后一个月。小蓝也在期盼着 2022 年的到来，那意味着小蓝又长大了一岁。你能帮小蓝计算一下还有几天就到 2022 年了吗？

输入：

两个整数 month，day 表示 2021 年的某月某日（0<month<13，0<day<32）

输出：

一个整数，表示距离 2020 年的天数。

样例输入：

12 31

样例输出：

1

```
#include <iostream>
using namespace std;
int main()
{
    int month,day,sum=0;
```

```
    cin>>month>>day;
    switch(month)
    {
        case 1:sum+=31;
        case 2:sum+=28;
        case 3:sum+=31;
        case 4:sum+=30;
        case 5:sum+=31;
        case 6:sum+=30;
        case 7:sum+=31;
        case 8:sum+=31;
        case 9:sum+=30;
        case 10:sum+=31;
        case 11:sum+=30;
        case 12:sum+=31;
    }
    sum-=day;
    sum++;
    cout<<sum;
    return 0;
}
```

3.2.3 大写字母 Y

自从小蓝学会编程之后，他酷爱用编程的方式写字。你能帮助他写出字母"Y"吗？

输入：整数 n（1（n<50）。

输出：输出高度为 2n-1 行的大写字母"Y"。

样例输入 1：

2

样例输出 1：

样例输入 2：

4

样例输出 2:

提示：第一行第一颗左侧无多余空格，每行最后一颗*后无多余空格。

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
    int t;
    for(int rr=1;rr<=n;rr++)
    {
        t=2*n-1-(rr-1);
        for(int cc= 1;cc<=rr-1;cc++)
        {
            cout<<" ";
        }
        for(int cc= rr ; cc<=t; cc++)
        {
            if(cc==rr || cc== t )
                cout<<"*";
            else
                cout<<" ";
        }

        cout<<endl;
    }
    for(int rr= n+1;rr<=(2*n-1);rr++)
    {
        for(int cc=1;cc<=n;cc++)
        {
            if(cc==n)
                cout<<"*";
            else
                cout<<" ";
        }
        cout<<endl;
    }
    return 0;}
```

3.2.4 “计算 24"

“计算 24"是一个流传已久的数字游戏，小蓝最近对此痴迷不已。

游戏规则是：对 4 个 1-10 之间的自然数，进行加、减、乘三种运算，要求运算結果等于 24。乘法的优先级高于加、减，并且算式中不可以用括号，不可以改变 4 个数字出现的顺序。

下面我们给出两个游戏的具体例子：

若给出的 4 个操作数是：10、2、4、8，则有两种可能的解答方案：

10+2+4+8=24，10*2-4+8=24，输出内容：2

若给出的 4 个操作数是：7、2、3.6，则没有解答案，输出内容：0。

输入：四个整数。

输出：输出方案总数

样例输入 1：

7 2 3 6

样例输出 1：

0

样例输入 2：

10 2 4 8

样例输出 2：

2

方法一：暴力枚举所有的可能性

```
#include <iostream>
using namespace std;
int a,b,c,d,f1,f2,f3,tmp;
int main( )
{
    int tj=0;
    cin>>a>>b>>c>>d;
    for(f1=1;f1<=3;f1++)
        for(f2=1;f2<=3;f2++)
            for(f3=1;f3<=3;f3++){
                if(f1==3&&f2==3&&f3==3) tmp=a*b*c*d;
                else if(f1==3&&f2==3&&f3!=3){
                    tmp=a*b*c;
                    if(f3==1)tmp+=d;
                    else tmp-=d;
                }
```

```
else if(f1!=3&&f2==3&&f3==3){
    tmp=b*c*d;
    if(f1==1)tmp=a+tmp;
    else tmp=a-tmp;
}
else if(f1==3&&f2!=3&&f3==3){
    if(f2==1)tmp=a*b+c*d;
    else tmp=a*b-c*d;
}
else if(f1==3&&f2!=3&&f3!=3){
    tmp=a*b;
    if(f2==1)tmp+=c;
    else tmp-=c;
    if(f3==1)tmp+=d;
    else tmp-=d;
}
else if(f1!=3&&f2==3&&f3!=3){
    tmp=b*c;
    if(f1==1)tmp=a+tmp;
    else tmp=a-tmp;
    if(f3==1)tmp+=d;
    else tmp-=d;
}
else if(f1!=3&&f2!=3&&f3==3){
    if(f1==1)tmp=a+b;
    else tmp=a-b;
    if(f2==1)tmp+=c*d;
    else tmp-=c*d;
}
else{
    if(f1==1)tmp=a+b;
    else tmp=a-b;
    if(f2==1)tmp+=c;
    else tmp-=c;
    if(f3==1)tmp+=d;
    else tmp-=d;
```

```
            }
            if(tmp==24)tj++;
        }
        cout<<tj;
    return 0;
}
```

方法二：递归 数组

```
#include<bits/stdc++.h>
using namespace std;
bool search(double a[],int x)  {
    if(x==1&&fabs(a[1]-24)<=1e-6)  return 1;  //题目要求除法运算为实数；
    double b[6];
    memset(b,0,sizeof(b));
    for(int i=1; i<x; ++i)
        for(int j=i+1; j<=x; ++j) {
            int p=1;
            for(int k=1; k<=x; ++k)
                if(k!=i&&k!=j)    b[p++]=a[k];
            b[p]=a[i]+a[j];
            if(search(b,x-1))return 1;//分别递归，依次穷举；
            b[p]=a[i]*a[j];
            if(search(b,x-1))return 1;
            b[p]=a[i]-a[j];
            if(search(b,x-1))return 1;
            b[p]=a[j]-a[i];
            if(search(b,x-1))return 1;
            if(a[j]!=0) {
                b[p]=a[i]/a[j];
                if(search(b,x-1))return 1;
            }
            if(a[i]!=0) {
                b[p]=a[j]/a[i];
                if(search(b,x-1))return 1;
            }
        }
```

```
    return 0;
}
int main() {
    while(1) {
        double a[6];
        memset(a,0,sizeof(a));
        int sum=0;
        for(int i=1; i<=4; ++i) {
            cin>>a[i];
            sum+=a[i];
        }
        if(sum==0) break;
        if(search(a,4))   cout<<"YES"<<endl;
        else cout<<"NO"<<endl;
    }
    return 0;
}
```

3.2.5 超级素数

在大于 1 的自然数中，除了 1 和它本身以外不再有其他因数的数，被称为素数，又叫质数。超级素数是指一个素数，每去掉最后一位上一个数字，总能保证剩下的数依然为素数。比如"373"就是一个超级素数，去掉个位的"3"后，"37"依然是素数：继续去掉"37"个位的"7"后，"3"还是素数。

输入：输人一个整数 n（10<=n<=10^8）

输出：输出所有小于等于 n 的超级素数的个数

样例输入 1：

30

样例输出 1：

6

样例输出 1 提示：2 3 5 7 23 29

样例输入 2：

50

样例输出 2：

8

样例输出 2 提示：2 3 5 7 23 29 31 37

算法思想

暴力枚举（70 分）

可以通过线性筛素数法将所有不超过 n 的素数求出，
然后枚举每个素数，判断是否符合超级素数的性质。
分析超级素数的性质，会发现最高位只能由素数 2 、 3 、 5 、 7 组成，
其余各位只能从是奇数 1 、 3 、 7 、 9 中选择，
因此可以使用 DFS 构造所有满足性质的超级素数。
*/

```
#include <iostream>
#include <cmath>
using namespace std;
int ans,n;
int a[]={2,3,5,7};//最高位
int b[]={1,3,7,9};//其他位
bool check(int x){//判断素数
    if(x==1)return false;
    for(int i=2;i<=sqrt(x);i++)
        if(x%i==0) return false;
    return true;
}
void dfs(int x){//深搜
    if(x>n)return;
    ans++;
    for(int i=0;i<4;i++)
        if(check(x*10+b[i]))
            dfs(x*10+b[i]);
}
int main( )
{
    cin>>n;
    for(int i=0;i<4;i++)
        dfs(a[i]);
    cout<<ans<<endl;
    return 0;
}
```

3.3***第十二届蓝桥青少组21年6月选拔赛c++高级组

一、选择题

1、表达式6-1的值是（ A）。（30分）

A.整数5　　B.字符5　　C.表达式不合法　　D.字符6

2、若二维数组a有n列，则在a[i][j]前元素个数为（B）。（30分）

A.i*n+j-1　　B.i*n+j　　C.j*n+i　　D.i*n+j+1

3、以下叙述中正确的是（ C）。（30分）

A.break语句只能用于switch语句体中。

B.continue语句的作用是：使程序的执行流程跳出包含它的所有循环。

C.break语句只能用在循环体内和switch语句体内。

D.在循环体内使用break语句和continue语句的作用相同。

4、按照“先进后出"原则组织数据的结构是（B）。（30分）

A.队列　　B.栈　　C.双向链表　　D.二叉树

5、用0、1、2、3、4这5个数字，能组成多少个没有重复数字的多位偶数？（　　）

A.144　　B.147　　C.160　　D.163　这一题没正确答案，正确答案是72种

3.3.1 双面打印

为了环保，妈妈一般都进行双面打印，也就是一张纸的正反面都打印出相应的内容。举个例子来说：如果一份电子材料有3页，那么需要2张纸进行打印；如果一份电子材料有4页，那么还是需要2张纸进行打印。现在已经知道了一份电子版的学习材料的页数N，你能帮小蓝计算一下需要几张纸吗？

输入样例：

7

输出样例：

4

```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
    cout<<(n+1)/2;//整型变量与整型参与运算结果还是整型
    return 0;
}
```

3.3.2 求完数

因子：因子也叫因数，例如 3×5=15，那么 3 和 5 是 15 的因子。同时 15×1=15，那么 1 和 15 也是 15 的因子。 1，3，5，15 这四个因子是 15 的所有因子。

完数：如果一个数等于不含它本身的其他因子之和，则称该数为‘完数’。如 6 的因子有 1，2，3，6，且 1+2+3= 6，因此 6 是完数。

题目描述: 输入一个正整数 N（0<N<10000）,输出小于 N 的所有完数及小于 N 的完数个数（个数前加“*”，例如：*2）。

```
#include<iostream>
using namespace std;
int main()
{
    int i,j,n,sum=0,count=0;
    cin>>n;
    for(i=1;i<n;i++)
    {
        for(j=1;j<i;j++)
        {
            if(i%j==0)
                sum=sum+j;
        }
        if(sum==i)
        {
            cout<<i<<endl;
            count++;
        }
        sum=0;
    }
    cout<<"*"<<count;
    return 0;
}
```

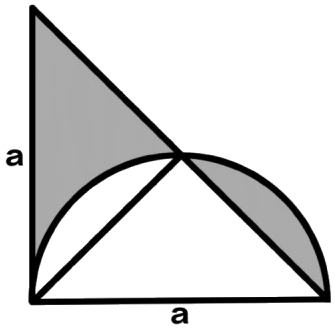
方法二：

```
#include <iostream>
using namespace std;
int wqs(int n)
{
```

```
 int i,sum=0;
 for(i=1;i<n;i++)
{
    if(n%i==0)  sum=sum+i;
}
if(sum==n) return 1;
else
return 0;
}

int main()
{
int  i,n,count=0;
cin>>n;
for(i=1;i<n;i++)
{
    if(wqs( i )==1)
    { count++;
     cout<<i<<endl;}
}
cout<<"*"<<count;
return 0;
}
```

3.3.3 求阴影部分面积

用户输入一个正整数 a（0<a<100），作为如图半圆的直径，同时作为如图等腰直角三角形的直角边长度，求下图的阴影部分面积，如下所示：



提示信息：三角形面积公式：S=（ah)/2（公式中 a 为三角形的底边，h 为底边所对应的高）

圆形面积公式：S=πr2（公式中 r 为圆的半径，π＝3.14）

已知条件：

1）半圆的直径和等腰直角三角形直角边长度相同；

2）三角形与半圆部分重叠；

输入描述

输入一个正整数 a（0<a<100）

输出描述

输出图形阴影面积（保留 2 位小数）

样例输入

10

样例输出

25.00

很简单，只要把半圆阴影移到三角那块就可以了，这样面积就是 a*a/2 了

只不过要保留两位小数，用 double

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
    double a;
    cin>>a;
    printf("%.2f",a*a/4);
    return 0;
}
```

3.3.4 判断分数（选择题判定）

选择题有 5 道，都是单选题，每道 30 分，共计 150 分。每道选择题选对得 30 分，选错或不选得 0 分。假设正确的答案为“DCBAD”，你能根据选手的提交情况，判定选手的选择题总分吗？

选手提交一个由 5 个字符组成的字符串，代表选手的选项。字符串仅能包含如下 5 种字符：“D”、“C”、“B”、“A”、“E”。其中“A”、“B”、“C”、“D”代表选手选择了某个选项，而“E”代表选手未做该题。求总分。

先定义一个字符串存放答案，输入选手的选项，通过一个循环和答案比对，只有答对的才加分，最后直接输出

```
#include <iostream>
using namespace std;
int main()
{
```

```
    string answer="DCBAD",n;
    int sum=0;
    cin>>n;
    for(int i=0; i<5; i++)
    {
        if (n[i]==answer[i])//读入的答案和标准答案逐位判断
            sum=sum+30;
    }
    cout<<sum;
    return 0;
}
```

3.3.5 节气

“二十四节气”被列入联合国教科文组织人类非物质文化遗产名录。在国际气象界，这一已有千年历史的时间认知体系被誉为“中国第五大发明”。

春雨惊春清谷天，夏满芒夏暑相连。秋处露秋寒霜降，冬雪雪冬小大寒。二十四节气，在四季轮回流淌，每个节气都有它较为稳定的日子。下表给出了农历庚子年（公历 2020 年 1 月 25 日~2021 年 2 月 11 日）中，二十四个节气的名称，公历具体日期及汉语拼音的缩写。

立春LC	2.4	雨水YS	2.19	惊蛰JZ	3.5	春分CF	3.20	清明QM	4.4	谷雨GY	4.19
立夏LX	5.5	小满XM	5.20	芒种MZ	6.5	夏至XZ	6.21	小暑XS	7.6	大暑DS	7.22
立秋LQ	8.7	处暑CS	8.22	白露BL	9.7	秋分QF	9.22	寒露HL	10.8	霜降SJ	10.23
立冬LD	11.7	小雪XX	11.22	大雪DX	12.7	冬至DZ	12.21	小寒XH	1.5	大寒DH	1.20

输入描述

整数 M，N（2≤M≤12,1≤N≤31），M，N 分别代表公历 2020 年的某月，某日。所给出的数据均为合法日期。

输出描述

如果当天恰好是一个节气，输出这个节气的汉语拼音缩写，如当天不是节气则输出下一个节气的汉语拼音缩写

先把月日和节气的节气缩写放在数组之中，先判断是否正好是节气，再进行判断最近的节气操作。

先判断是否是 12 月的边界特例，再判断其他情况

其他情况有两种：

日期超过了当前遍历的节气，直接 continue

日期没有超过，直接输出

```
#include<iostream>
#include<cstring>
using namespace std;
int main() {
    int x,y;
    int month[24]= {1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,12,12};
    int day[24]=
{5,20,3,18,5,20,4,19,5,20,5,21,6,22,7,22,7,22,8,23,7,22,7,21};
    string jq[24]=
{"XH","DH","LC","YS","JZ","CF","QM","GY","LX","XM","MZ","XZ","XS","DS","L
Q","CS","BL","QF","HL","SJ","LD","XX","DX","DZ"};
    cin>>x>>y;
    for (int i=0; i<24; i++)
    {
        if (month[i]==x&&day[i]==y)
        {
            cout<<jq[i];
            return 0;
        }
    }

    for (int i=0; i<24; i++)
    {
        if (x==12&&y>day[23])       //12月末特例
        {
            cout<<jq[0];
            return 0;
        }
        else if (x>month[i]||y>day[i])
            continue;                   //未到时间
        else if (x<=month[i]&&y<day[i])
        {                       //寻找
            cout<<jq[i];
            return 0;
        }
    } }
```

方法 2

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
    int x,y;
    int month[24]= {1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,12,12};
    int day[24]=
{5,20,3,18,5,20,4,19,5,20,5,21,6,22,7,22,7,22,8,23,7,22,6,21};
    string jq[24]=
{"XH","DH","LC","YS","JZ","CF","QM","GY","LX","XM","MZ","XZ","XS","DS","L
Q","CS","BL","QF","HL","SJ","LD","XX","DX","DZ"};
    cin>>x>>y;
    int t;
    for (int i=0; i<24; i++)
    {
        if (month[i]==x&&day[i]==y)
        {
            cout<<jq[i];
            return 0;

        }
    }
    if ( (x==12&&y>day[23]) ||  (x==1 && y<day[0])  )       //12 月末特例
    {
        cout<<jq[0];
    }

    for (int i=1; i<24; i++)
    {
        if (x>month[i] || y>day[i] ) continue;
        else if (x<=month[i]&&  y<day[i] )
        {
            cout<<jq[i];
        }
```

```
        return 0;
    }

}
```

3.3.6 垃圾分类

不知北京的小同学们会不会扔垃圾啊？

2019 年 7 月 1 日，上海正式实施垃圾分类。

2020 年 5 月 1 日，北京正式实施垃圾分类。

不要以为垃圾分类这件事离自己很远，到 2020 年底，全国将有 46 个重点城市要基本建成垃圾分类处理系统。

小伙伴们，你们准备好了吗？我们就以《北京市生活垃圾分类管理条例》为参考，学习一下垃圾分类的知识吧。

生活垃圾基本分类有四种：厨余垃圾，可回收物，有害垃圾和其他垃圾。

1."厨余垃圾"指家庭中产生的菜帮菜叶、瓜果皮核、剩菜剩饭、废弃食物等易腐垃圾。

2."可回收物”指在日常生活中已失去原有使用价值，但回收后加工可再利用的垃圾；

3."有害垃圾"指生活垃圾中的有毒有害物质；

4."其他垃圾"指除上述垃圾之外的生活垃圾，及难辨识类别的生活垃圾。

下表中列举了四类垃圾中的三个常见例子，及其英文标识。

1.厨余垃圾（FOOD WASTE）：菜叶（leaves），西瓜皮（watermelon peel），剩饭（leftovers）.

2.可回收物（RECYCLABLE）：纸箱（paper box）.塑料瓶（plastic botte）衣服（clothes）

3.有害垃圾（HAZARDOUS）：充电电池（rechargeable battery），弃置药品（abandoned medicine）

消毒剂（disinfectant）

4.其他垃圾（RESIDUAL WASTE）：口罩（mask），普通电池（battery）.塑料袋（plastic bag）.

学会了吗？现在请你根据输入的（1-20）个垃圾名称，将这些垃圾进行归类统计，并将出现次数最多的垃圾种类的英文名称，及其出现次数打印出来。

输入描述

第一行为一个正整数 N(1<=N<=20）代表接下来将出现的垃圾行数。

接下来 N 行，每行是上表中某一种具体的垃圾的英文名称。

（注意：测试数据中，出现次数最多的垃圾种类只有一种）

输出描述

输出结果有 2 行。

第一行是出现次数最多的垃圾种类的英文名称。第二行是出现次数最多的垃圾种类的出现次数。

样例输入

5

leaves

mask

disinfectant

leftovers

watermelon

peel

样例输出

FOOD WASTE

3

提示

样例说明：出现的 5 行垃圾中，厨余垃圾（FOOD WASTE）出现 3 次，依次是 leaves/leftovers/watermelon peel；别的垃圾种类只出现了 2 次。

所以厨余垃圾（FOOD WASTE）出现次数最多，为 3 次。

3.3.7 成绩统计

小蓝给学生们组织了一场考试，卷面总分为 100 分，每个学生的得分都是一个 0 到 100 的整数。

如果得分至少是 60 分，则称为及格。如果得分至少为 85 分，则称为优秀。

请计算及格率和优秀率，用百分数表示，百分号前的部分四舍五入保留整数。

输入格式

输入的第一行包含一个整数 n，表示考试人数。

接下来 n 行，每行包含一个 0 至 100 的整数，表示一个学生的得分。

输出格式

输出两行，每行一个百分数，分别表示及格率和优秀率。百分号前的部分四舍五入保留整数。

样例输入

7

80

92

56

74
88
100
0
样例输出
71%
43%
参考答案：

```
#include<iostream>
using namespace std;
int main()
{
    double a1=0,a2=0;
    int n,i,score;
    cin>>n;
    for(i=1; i<=n; i++)
    {
        cin>>score;
        if(score>=60) a1++;
        if(score>=85) a2++;
    }
    a1=int((a1/n+0.005)*100);
    a2=int((a2/n+0.005)*100);
    cout<<a1<<"%"<<endl;
    cout<<a2<<"%";
    return 0;
}
```

3.3.8 杨辉三角

杨辉三角又称作中国三角形

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1

1　　7　　21　　35　　35　　21　　7　　1

请在观察杨辉三角的规律后,编写一段程序输出当输入 n 时,输出杨辉三角的前 n 行

例:输入 3

输出:

1

1　　1

1　　2　　1

解析：（1）第一列和对角线上的元素都为 1；

（2）除第一列和对角线上的元素之外，其他的元素的值均为前一行上的同列元素和前一列元素之和。

方法一：数组法

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

int main()
{
    int n,i,j;
    cin>>n;
    int a[n][n];
    for(i=0; i<=n-1; i++)
    {
        a[i][0]=1;
        a[i][i]=1;
    }

    for(i=2; i<=n-1; i++)
    {
        for(j=1; j<=i-1; j++)
        {
            a[i][j]=a[i-1][j-1]+a[i-1][j];//每个数是上面两数之和
        }
        cout<<endl;
    }
```

```
    for(i=0; i<=n-1; i++)
    {
        for(j=0; j<=i; j++)
        {
            printf("%6d",a[i][j]);
        }
        cout<<endl;
    }
    return 0;
}
```

方法二、迭代公式法

```
#include <iostream>
#include<iomanip>
using namespace std;
int main() {
    int n,p;
    cin>>n;
    for(int i=1; i<=n; i++)
    {
        p=1;
        cout<<p;//输出每一个列的第一个数 1
        for(int j=1; j<=i; j++)
        {
            p=p*(i-j)/j; //杨辉三角相邻两个数的迭代公式
            cout<<setw(6)<<p;
        }
        cout<<endl;
    }
    return 0;
}
```

3.3.9 所有三位数水仙花数

水仙花数（Narcissistic number）也被称为超完全数字不变数（pluperfect digital invariant, PPDI）、自恋数、自幂数、阿姆斯壮数或阿姆斯特朗数（Armstrong number），水仙花数是指一个 3 位数，它的每个位上的数字的 3 次幂之和等于它本身（例如：1^3 + 5^3+ 3^3 = 153）。

编程输出所有的三位数水仙花数。

```
#include <iostream>
using namespace std;
int main()
{
    int a,b,c,i;
    for(i=100; i<1000; i++)
    {
        a=i%10;
        b=i/10%10;
        c=i/100%10;
        if (a*a*a+b*b*b+c*c*c==i)
            cout<<i<<"  ";
    }
    return 0;
}
```

3.3.10判断某个年份的生肖（非常经典的解法）

已知 1900 年是鼠年，输入一个年份，输出其对应生肖。

鼠 rat 牛 ox 虎 tiger 兔 rabbit 龙 dragon 蛇 snake

马 horse 羊 sheep 猴 monkey 鸡 rooster 狗 dog 猪 pig

输入：

输入一个整数 year 表示年份 （1≤year≤9999）。

输出：

输出一个单词，表示生肖。

```
//鼠 rat 牛 ox 虎 tiger 兔 rabbit 龙 dragon 蛇 snake
//马 horse 羊 sheep 猴 monkey 鸡 rooster 狗 dog 猪 pig
#include <iostream>
using namespace std;
int main()
{
    char
p[12][8]={"rat","ox","tiger","rabbit","dragon","snake","horse","sheep","m
onkey","rooster","dog","pig"};
    int year0=1900;   //1900 年是鼠年，作为基准
    int year,n;
    cin>>year;
    n=(year-year0)%12; //输入年份-基准年份  算周期余数
```

```
    if (n<0)
    {
       n=n+12;
    }
    cout<<p[n]<<endl;
}
```

方法 2

```
#include <iostream>
using namespace std;
int main()
{
    char
p[12][8]={"rat","ox","tiger","rabbit","dragon","snake","horse","sheep","m
onkey","rooster","dog","pig"};
    int year0=1900;   //1900 年是鼠年，作为基准
    int year,n;
    cin>>year;
    if (year>=year0)
        n=(year-year0)%12; //输入年份-基准年份  算周期余数
    else
       n = (year0-year)%12;

    cout<<p[n]<<endl;
}
```

方法 3

```
#include <iostream>
using namespace std;
int main()
{
    string
p[12]={"rat","ox","tiger","rabbit","dragon","snake","horse","sheep","monk
ey","rooster","dog","pig"};
    int year0=1900;   //1900 年是鼠年，作为基准
    int year,n;
```

```
    cin>>year;
    if (year>=year0)
       n=(year-year0)%12; //输入年份-基准年份  算周期余数
    else
       n = (year0-year)%12;

    cout<<p[n]<<endl;
}
```

3.3.11 Fibonacci 斐波那契数列

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。

输入格式 输入包含一个整数 n。

输出格式 输出一行，包含一个整数，表示 Fn 除以 10007 的余数。

说明：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。

样例输入 10 样例输出 55

样例输入 22 样例输出 7704

数据规模与约定 1 <= n <= 1,000,000

```
#include<iostream>
using namespace std;
int main()
{
    long long f1=1,f2=1,f3;
    int n,s;
    cin>>n;
     if(n>2)
    for(s=3;s<=n;s++)
    {
         f3=(f2+f1)%10007;
         f1=f2;
         f2=f3;
     }
     cout<<f3;
     return 0;
 }
```

3.3.12递归法取数位

求 1 个整数的第 k 位数字有很多种方法。

以下的方法就是一种。

// 求 x 用 10 进制表示时的数位长度

```
#include<iostream>
#include <cstdio>
using namespace std;
int len(int x)
{
    if(x<10) return 1;
    return len(x/10)+1;
}
// 取 x 的第 k 位数字
int f(int x, int k)
{
    if(len(x)-k==0) return x%10;
    return f(x/10,k);
}
int main()
{
    int x = 23574;
    printf("%d\n", f(x,3));
    return 0;
}
```

3.3.13图片旋转

图片旋转是对图片最简单的处理方式之一，在本题中，你需要对图片顺时针旋转 90 度。我们用一个 $n \times m$ 的二维数组来表示一个图片，例如下面给出一个 3×4 的图片的例子：

3 4

1 3 5 7

9 8 7 6

3 5 9 7

这个图片顺时针旋转 90 度后的图片如下：

3 9 1

5 8 3

9 7 5

7 6 7

给定初始图片，请计算旋转后的图片。

输入格式

输入的第一行包含两个整数 n 和 m，分别表示行数和列数。

接下来 n 行，每行 m 个整数，表示给定的图片。图片中的每个元素（像素）为一个值为 0 至 255 之间的整数（包含 0 和 255）。

输出格式

输出 m 行 n 列，表示旋转后的图片。

样例输入

3 4

1 3 5 7

9 8 7 6

3 5 9 7

样例输出

3 9 1

5 8 3

9 7 5

7 6 7

```
#include<iostream>
using namespace std;
int main()
{
    int n, m;
    cin >> n >> m;
    int map[200][200];
    for (int i = 0; i < n; i++) //读取并存储矩阵
    {
        for (int j = 0; j < m; j++)
            cin >> map[i][j];
    }
    for (int i = 0; i < m; i++)
    { //翻转并输出矩阵
        for (int j = n - 1; j >= 0; j--)
        {
            cout << map[j][i]<<" ";
        }
```

```
        cout<<endl;
    }
    return 0;
}
```

3.3.14第 2019 个质数

我们知道第一个质数是 2 第二个质数是 3 第三个质数是 5.....请你计算第 2019 个质数是多少?

```
#include<iostream>
#include<cmath>
using namespace std;
int zs(int a)
{
    int n=sqrt(a);
    for (int i = 2; i <=n ; i++)
    if (a % i == 0)
        return 0;
    return 1;
}
int main()
{
    int cnt = 0,i=1;
    while(cnt<2019)
    {
        i++;
        if (zs(i) == 1)
            cnt++;
    }
    cout << i ;
    return 0;
}
    /*
    答案：17569
    */
```

3.3.15年号字串(蓝桥杯 2019 选拔赛)

小明用字母 A 对应数字 1，B 对应 2，以此类推，用 Z 对应 26。对于 27 以上的数字，小明用两位或更长位的字符串来对应，例如 AA 对应 27，AB 对 应 28，

AZ 对应 52，LQ 对应 329。

请问 2019 对应的字符串是什么？

解题思路：可以将它看成二十六进制。

2019 % 26 = 77 余 17 ，17 对应 Q

77 % 26 = 2 余 25，25 对应 Y

2 % 26 = 0 余 2,2 对应 B

答案：BYQ

```
#include<iostream>
using namespace std;
int main()
{
    int m = 2019;
    while( m!=0)
    {
        int t = m%26;
        m=m/26;
        cout << char(t+64);
    }
    return 0;
}
```

//正确答案是输出结果的倒序

输出：QYB

答案：BYQ

3.3.16矩形切割

小明有一些矩形的材料，他要从这些矩形材料中切割出一些正方形。

当他面对一块矩形材料时，他总是从中间切割一刀，切出一块最大的正方 形，剩下一块矩形，然后再切割剩下的矩形材料，直到全部切为正方形为止。

例如，对于一块两边分别为 5 和 3 的材料（记为 5×3），小明会依次切出 3×3、2×2、1×1、1×1 共 4 个正方形。

现在小明有一块矩形的材料，两边长分别是 2019 和 324。请问小明最终会 切出多少个正方形？

```
#include <bits/stdc++.h>
using namespace std;
int work(int a, int b)
{
    if (a == b)
```

```
        return 1;
    int k = min(a, b);
    int l = max(a, b);
    return work(k, l - k) + 1;
}
int main()
{
    int a = 2019, b = 324;
    cout << work(a, b) ;
    return 0;
}//答案：21
```

3.4 ***第十届蓝桥杯青少组选拔赛 C++中级组（2019 年 12 月）

3.4.1 分段输出

输入一个正整数 N，如果 N 大于等于 90 输出 A，如果 N 大于等于 80 且小于 90 输出 B，如果 N 大于等于 70 且小于 90 输出 C，如果 N 小于 70 输出 D。

输入

70

输出

C

```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
    if(n>=90)        cout <<"A";
    if(n>=80&&<90) cout <<"B";
    if(n>=70&&<90) cout <<"C";
    if(n<70)        cout <<"D";
    }
    return 0;
}
```

3.4.2 小球反弹高度

一个小球从 n 米高度自由落下，每次落地后反跳回原高度的一半，再落下。求第 10 次反弹多高，及初始落下到第 10 次反弹到最高点时（不含第十次落下距离）一共经历了多少米？

输入描述：输入一个正整数 n

输出描述：（1）第 10 次反弹高度 （2）一共经历了多少米（不含第十次落下距离）

输入

1024

输出

1

```
#include<iostream>
using namespace std;
int main()
{
double n,i,sum=0;
cin>>n;
sum=n;
for(i=1;i<=10;i++)
{
sum=sum+n;
n=n/2;
}
cout<<n<<endl<<sum-n;
return 0;
}
```

3.4.3 求最大值、最小值和平均值

输入 10 个正整数，以空格分隔，输出其中的最大值、最小值以及平均值，以逗号隔开

输入

1 2 3 4 5 6 7 8 9 10

输出

10,1,5.5

```
#include<iostream>
using namespace std;
int main()
```

```
{
int n,max,min,sum=0;
cin>>n;
max=min=n;
for(int i=2;i<=10;i++)
{
cin>>n;
if(n>max) max=n;
if(n<min) min=n;
sum=sum+n;
}
cout<<max<<","<<min<<","<<sum/10.0;
return 0;
}
```

3.4.4 回文数

输入 n，输出 1~n（包含 1 和 n）之间所有的回文数，并统计回文数的个数。个数之前要加星号。

输入：

100

输出：

1

2

3

4

5

6

7

8

9

11

22

33

44

55

66

77

```
    88
    99
    *18
#include<iostream>
using namespace std;
int main() {
    int i,n,count=0,t,k=10;
    cin>>n;
    for( i=1; i<=n; i++)
    {
        t=i;
        int m=0;
        while(t!=0)
        {
            m=m*k+t%10;
            t=t/10;
        }
        if(m==i)
        {
            cout<<m<<endl;
            count++;
        }
    }
    cout<<"*"<<count;
    return 0;
}
```

3.4.5 煤球数目

有一堆煤球，堆成三角棱锥形。具体：

第一层放 1 个，

第二层 3 个（排列成三角形），

第三层 6 个（排列成三角形），

第四层 10 个（排列成三角形），

…

如果一共有 100 层，共有多少个煤球？

请输出表示煤球总数目的数字。

注意：你输出的应该是一个整数，不要输出任何多余的内容或说明性文字。

输入

没有输入。

输出

输出一个整数，即 1~100 层煤球的总数。

解题思路：第一层 1

第二层 1+2=3

第三层 3+3=6

第四层 6+4=10

……

第 i 行煤球的个数是在原有结果的基础上加上第 i 行的 i 的值

答案：171700

方法一、

```
#include<iostream>
using namespace std;
int main ()
{
    int i,temp = 0;
    long long sum = 0;
    for(i=1;i<=100;i++)
    {
        temp=temp+i; //每层递增数量和层数相同
        sum=sum+temp;
    }
    cout<<sum;
    return 0;
 }
```

方法二、观察发现第 n 层有(n*(n+1))/2 个煤球，只需要使用一个循环将每一层的数量累加起来即可。

```
#include<iostream>
using namespace std;
int main ()
{
    long long sum = 0;
    for(int i=1;i<=100;i++)
        sum=sum+(i*(i+1))/2;  //第 n 层有(n*(n+1))/2 个煤球
    cout<<sum;
```

```
    return 0;
} //比较精简的代码
```

3.4.6 奖券数目

有些人很迷信数字，比如带“4”的数字，认为和“死”谐音，就觉得不吉利。虽然这些说法纯属无稽之谈，但有时还要迎合大众的需求。某抽奖活动的奖券号码是 5 位数（10000-99999），要求其中不要出现带“4”的号码，主办单位请你计算一下，如果任何两张奖券不重号，最多可发出奖券多少张。

请提交该数字（一个整数），不要写任何多余的内容或说明性文字。

```
int main() {
    int ans = 0;
    for (int i = 10000; i <= 99999; i++) {
        int num = i;
        int a = num % 10;     num=num/10;
        int b = num % 10;    num=num/10;
        int c = num % 10; num=num/10;
        int d = num % 10; num=num/10;
        int e = num % 10;
        if(a==4||b==4||c==4||d==4||e==4)
        {     }
        else
        ans++;
    }
    cout <<ans;
    return 0; }
```

3.4.7 纪念日

2020 年 7 月 1 日是中国共产党成立 99 周年纪念日。

中国共产党成立于 1921 年 7 月 23 日。

请问从 1921 年 7 月 23 日中午 12 时到 2020 年 7 月 1 日中午 12 时一共包含多少分钟？

本题的结果为一个整数，在提交答案时只填写这个整数，填写多余的内容将无法得分。

```
#include<iostream>
using namespace std;
int days(int n) {
    if(n%400==0||n%4==0&&n%100!=0)
```

```
    return 366;
    else
    return 365;
}
int main() {
    int sum=0;
    for(int i=1922; i<=2020; i++)
    {sum=sum+days(i);
       }
    sum=sum-22;
    cout<<sum*60*24;
    return 0; }
```

3.4.8 生日蜡烛

某君从某年开始每年都举办一次生日 party，并且每次都要吹熄与年龄相同根数的蜡烛。现在算起来，他一共吹熄了 236 根蜡烛。请问，他从多少岁开始过生日 party 的？

答案：26

```
#include<stdio.h>
int main()
{
    int start,end;
    for(start = 1 ; start < 236 ; start ++)
    {
        for( end = start ; end < 236 ; end ++ )
        {
            int sum = 0;
            for(int i = start; i <= end; i ++)
                sum += i;
            if( sum == 236)
            {
                printf("start : %d end : %d\n",start,end);
            }
        }
    }
    return 0;
```

}

3.5***第十三届蓝桥杯青少年 C++中级 11 月选拔赛

3.5.1 A+B 问题

给定两个整数 A 和 B，输出 A+B 的值。保证 A、B 及结果均在整型范围内。现在请你解决这一问题。
输入描述： 一行，包含两个整数 A，B，中间用单个空格隔开。A 和 B 均在整型范围内。
输出描述：一个整数，即 A+B 的值。保证结果在整型范围内。
样例输入：1 2
样例输出：3

```
#include<iostream>
using namespace std;
int main()
{
    long long a,b; //注意数据范围 尽量用 long long
    cin>>a>>b;
    cout << a+b << endl;
    return 0;
}
```

3.5.2 统计数字

输入一段英文（包含英文字母和数字），统计出这段字符串共有多少个数字字符。输入描述：输入一段英文字符串（字符串长度< 3 * 10^4）。
输出描述：数字字符的个数
样例输入：a1b2c3
样例输出：3
方法一：字符数组

```
#include<iostream>
#include<cstring> //strlen( ) 函数使用此文件头
using namespace std;
char s[30010];
int cnt;
```

```
int main()
{
    cin >>s;//遇到空格或者换行符就停止
    int len=strlen(s);//求长度，strlen== string length
    for(int i= 0; i < len; i++)
        if(s[i] >='0'&& s[i]<= '9')
            cnt++;
    cout << cnt << endl;
    return 0;
}
```

方法二：声明字符串

```
#include<iostream>
#include<cstring> //strlen( ) 函数使用此文件头
using namespace std;
string s;
int cnt;
int main()
{
    cin >>s;//遇到空格或者换行符就停止
    int len=s.length();//求长度 s.size()
    for(int i= 0; i < len; i++)
        if(s[i] >='0'&& s[i]<= '9')
            cnt++;
    cout << cnt << endl;
    return 0;
}
```

3.5.3 班级排列

要拍一个班级照，现在要让男生从低到高排列，女生从高到低排列。输入 10 个学生的身高信息，然后下一排输入男生女生的信息，男生是 1，女生是 2.

输入描述：第一排输入 10 个学生的身高，带小数;第二排输入 10 个男女生信息，男生为 1，女生为 2;

输出描述：按要求排序后的身高，用空格隔开，身高保留两位小数。

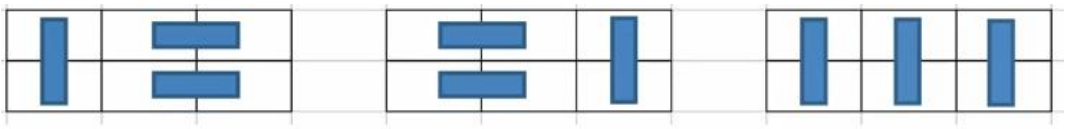
样例输入：

1.58 1.70 1.71 1.63 1.81 1.55 1.50 1.53 1.74 1.61
2 1 1 2 1 2 2 2 1 1
样例输出：
1.61 1.70 1.71 1.74 1.81 1.63 1.58 1.55 1.53 1.50
解题思路：模拟法 声明三个数组 分别存储男女混合临时身高、男生身高和女生身高 读入身高数据 并根据男生还是女生存储到相应数组里 使用数组排序 然后输出。

```
#include<iostream>
#include<algorithm>
using namespace std;
int b=0,g=0,i,t;
float boy[10]={},girl[10]={}; //初始化数组为0
float temp[10]={}; //存放读入的数据
int main()
{  for(i=0;i<10;i++)
     cin>>temp[i];
   for(i=0;i<10;i++)
    {
    cin>>t;
    if(t==1)
    { boy[b]=temp[i]; b++; }
     if(t==2)
    { girl[g]=temp[i]; g++;}

    }
    sort(boy,boy+b); //数组排序函数 sort()默认升序
    sort(girl,girl+g);
   for(int i= 0; i < b; i++)
       printf("%0.2f ",boy[i]);  //要求保留两位小数注意避坑
   for(int i= 0; i < g; i++)
       printf("%0.2f ",girl[g-1-i]); //倒序输出
    return 0;
}
```

3.5.4 铺地砖

在 2*n 的一个长方形方格，用一个 1*2 的骨牌铺满方格。



编写一个程序，试对给出的任意一个 n（n>0），输出铺法总数。
输入描述：输入 n，n 小于等于 20
输出描述：输出铺法总数
样例输入：3
样例输出：3
解题分析:用模拟的方法 计算 n 等于 1 2 3 4 5 6 的情况 会发现输出结果符合斐波那契数列的规律。然后用递归或者枚举方式解题即可。

```
#include<iostream>
using namespace std;
long long a[30]={0,1,2};
int n;
int main()
{
cin >>n;
for(int i=3;i<=n; i++)
    a[i]= a[i-1]+ a[i-2];
//for(int i=1;i<=n; i++)
//  cout<<a[i]<<" ";  用于测试看是否是斐波那契数列
cout<<a[n]<<endl;
 return 0;
}
```

3.5.5 新冠病毒

疫情期间，有些人打了疫苗，在表格中表示为 0，有些人感染了病毒，在表格中表示为 X，还有些人没有打疫苗，在表格中表示为 P。
新冠病毒会传染，没有打疫苗的人都会被传染，有病毒的人 1 分钟会传染周围上下左右的 4 个人。最后输出有多少人感染了病毒。打了疫苗的人不会被感染。
输入描述：第一行输入 2 个正整数 n 和 m（2<n，m<30）表示一个 n*n 的表格，过了 m

分钟的时间。
接下来 n 行，每行 n 个字母，仅包含 O、P、X，分别表示打了疫苗的人，没打疫苗的人，有新冠病毒的人。
输出描述：输出一个数字，表示 m 分钟后会有多少人感染新冠病毒。
样例输入：4 3
P P P X
P O P P
P P P P
P O P X
样例输出：11

```
#include<iostream>
#include<cstring> //memset( )函数使用
using namespace std;
char a[40][40];
int n, m, v[40][40];
int main() {
    cin >> n >> m; //n*n 的矩阵，m 轮
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            cin >> a[i][j]; //进行 m 轮的传染
    while (m--) {
        memset(v, 0, sizeof(v));
//将 v 所指向的某一块内存中的每个字节的内容全部设置为 ch 指定的 ASCII 值
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++) {
                if (a[i][j] == 'X' && v[i][j])
                    if (a[i - 1][j] == 'P') {
                        a[i - 1][j] = 'X';
                        v[i - 1][j] = 1;
                    }
                if (a[i + 1][j] == 'P') {
                    a[i + 1][j] = 'X';
                    v[i + 1][j] = 1;
                }
                if (a[i][j + 1] == 'P') {
                    a[i][j + 1] = 'X';
```

```
                v[i][j + 1] = 1;
            }
            if (a[i][j - 1] == 'P') {
                a[i][j - 1] = 'X';
                v[i][j - 1] = 1;
            }
        }
    }
int cnt = 0;
for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j++)
        if (a[i][j] == 'X')
            cnt++;
cout << cnt << endl;
return 0;
}
```

3.6 ****蓝桥杯官网练习系统基础题目汇总（选拔赛省赛必看）

3.6.1 实数输出

给定圆的半径 r，求圆的面积。

输入格式

输入包含一个整数 r，表示圆的半径。

输出格式

输出一行，包含一个实数，四舍五入保留小数点后 7 位，表示圆的面积。

说明：在本题中，输入是一个整数，但是输出是一个实数。

对于实数输出的问题，请一定看清楚实数输出的要求，比如本题中要求保留小数点后 7 位，则你的程序必须严格的输出 7 位小数，输出过多或者过少的小数位数都是不行的，都会被认为错误。

实数输出的问题如果没有特别说明，舍入都是按四舍五入进行。

样例输入

4

样例输出

50.2654825

数据规模与约定

1 <= r <= 10000。

本题对精度要求较高，请注意π的值应该取较精确的值。你可以使用常量来表示π，比如 PI=3.14159265358979323，也可以使用数学公式来求π，比如 PI=atan(1.0)*4。

```
#include<iostream>
#include <stdio.h>
#include <math.h>
using namespace std;
int main()
{
    int r;
    double s, PI;
    cin>>r;
    PI = atan(1.0) * 4;
    s = PI * r * r;
    printf("%0.7lf", s);
    return 0;
}
```

3.6.2 求和公式

求 1+2+3+...+n 的值。

输入格式

输入包括一个整数 n。

输出格式

输出一行，包括一个整数，表示 1+2+3+...+n 的值。

样例输入

4

样例输出

10

样例输入

100

说明：有一些试题会给出多组样例输入输出以帮助你更好的做题。

一般在提交之前所有这些样例都需要测试通过才行，但这不代表这几组样例数据都正确了你的程序就是完全正确的，潜在的错误可能仍然导致你的得分较低。

样例输出

5050

数据规模与约定

1 <= n <= 1,000,000,000。

说明：请注意这里的数据规模。

本题直接的想法是直接使用一个循环来累加，然而，当数据规模很大时，这种“暴力”的方法往往会导致超时。此时你需要想想其他方法。你可以试一试，如果使用1000000000作为你的程序的输入，你的程序是不是能在规定的上面规定的时限内运行出来。

本题另一个要值得注意的地方是答案的大小不在你的语言默认的整型(int)范围内，如果使用整型来保存结果，会导致结果错误。

如果你使用C++或C语言而且准备使用printf输出结果，则你的格式字符串应该写成%I64d以输出long long类型的整数。

等差数列求和公式：首项加末项乘以项数除以2　(1+n)*n/2

```
#include<iostream>
using namespace std;
int main()
{
    long long n;
    cin>>n;
    cout<<(1+n)*n/2;
    return 0;
}
```

3.6.3 A+B 问题

输入A、B，输出A+B。

说明：在“”这部分，会给出试题的意思，以及所要求的目标。

输入格式

输入的第一行包括两个整数，由空格分隔，分别表示A、B。

说明：“输入格式”是描述在测试你的程序时，所给的输入一定满足的格式。

做题时你应该假设所给的输入是一定满足输入格式的要求的，所以你不需要对输入的格式进行检查。多余的格式检查可能会适得其反，使用你的程序错误。

在测试的时候，系统会自动将输入数据输入到你的程序中，你不能给任何提示。比如，你在输入的时候提示“请输入A、B”之类的话是不需要的，这些多余的输出会使得你的程序被判定为错误。

输出格式

输出一行，包括一个整数，表示A+B的值。

说明：“输出格式”是要求你的程序在输出结果的时候必须满足的格式。

在输出时，你的程序必须满足这个格式的要求，不能少任何内容，也不能多任何内容。如果你的内容和输出格式要求的不一样，你的程序会被判断为错误，包括你输出了提示信息、中间调试信息、计时或者统计的信息等。

样例输入

12 45

说明：“样例输入”给出了一组满足“输入格式”要求的输入的例子。

这里给出的输入只是可能用来测试你的程序的一个输入，在测试的时候，还会有更多的输入用来测试你的程序。

样例输出

57

说明：“样例输出”给出了一组满足“输出格式”要求的输出的例子。

样例输出中的结果是和样例输入中的是对应的，因此，你可以使用样例的输入输出简单的检查你的程序。

比如，对于本题，如果你写一个程序不管输入是什么都输入 57，则样例数据是对的，但是测试其他数据，哪怕输入是 1 和 2，这个程序也输出 57，则对于其他数据这个程序都不正确。

数据规模与约定

-10000 <= A, B <= 10000。

说明：“数据规模与约定”中给出了试题中主要参数的范围。

这个范围对于解题非常重要，不同的数据范围会导致试题需要使用不同的解法来解决。比如本题中给的 A、B 范围不大，可以使用整型(int)来保存，如果范围更大，超过 int 的范围，则要考虑其他方法来保存大数。

有一些范围在方便的时候是在“”中直接给的，所以在做题时不仅要看这个范围，还要注意。

```
#include<iostream>
using namespace std;
int main()
{
    int a, b;
    cin >> a >> b;
    cout << a + b
 return 0;
}
```

3.6.4 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。

其他的年份都不是闰年。

输入格式

输入包含一个整数 y，表示当前的年份。

输出格式

输出一行，如果给定的年份是闰年，则输出 yes，否则输出 no。

说明：当试题指定你输出一个字符串作为结果（比如本题的 yes 或者 no，你需要严格按照试题中给定的大小写，写错大小写将不得分。

样例输入

2013

样例输出

no

样例输入

2016

样例输出

yes

数据规模与约定

1990 <= y <= 2050。

```
#include<iostream>
using namespace std;
int main()
 {
    int y;
    cin >> y;
    if (y%4==0 && y%100!=0 || y%400==0)
        cout << "yes" << endl;
    else
        cout << "no" << endl;
    return 0;
}
```

3.6.5 01 字串

对于长度为 5 位的一个 01 串，每一位都可能是 0 或 1，一共有 32 种可能。它们的前几个是：

00000

00001

00010

00011

00100

请按从小到大的顺序输出这 32 种 01 串。

输入格式

本试题没有输入。

输出格式

输出 32 行，按从小到大的顺序每行一个长度为 5 的 01 串。

样例输出

00000

00001

00010

00011<以下部分省略>

```
#include<iostream>
using namespace std;
int main()
{
    for(int i=0; i<=1; i++)
        for(int j=0; j<=1; j++)
            for(int k=0; k<=1; k++)
                for(int l=0; l<=1; l++)
                    for(int m=0; m<=1; m++)
                        cout<<i<<j<<k<<l<<m<<endl;
    return 0;
}
```

3.6.6 字母图形

利用字母可以组成一些美丽的图形，下面给出了一个例子：

ABCDEFG

BABCDEF

CBABCDE

DCBABCD

EDCBABC

这是一个 5 行 7 列的图形，请找出这个图形的规律，并输出一个 n 行 m 列的图形。

输入格式

输入一行，包含两个整数 n 和 m，分别表示你要输出的图形的行数的列数。

输出格式

输出 n 行，每个 m 个字符，为你的图形。

样例输入

5 7

样例输出

ABCDEFG

BABCDEF

CBABCDE

DCBABCD

EDCBABC

数据规模与约定：1 <= n, m <= 26。

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int n, m;
    cin >> n >> m;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j)
            cout << char('A'+abs(i-j));
        cout << endl;
    }
    return 0;
}
```

3.6.7 数列特征

给出 n 个数，找出这 n 个数的最大值，最小值，和。

输入格式

第一行为整数 n，表示数的个数。

第二行有 n 个数，为给定的 n 个数，每个数的绝对值都小于 10000。

输出格式

输出三行，每行一个整数。第一行表示这些数中的最大值，第二行表示这些数中的最小值，第三行表示这些数的和。

样例输入

5

1 3 -2 4 5

样例输出

5 -2 3

数据规模与约定

1 <= n <=。

```
#include<iostream>
```

```
#include<cstring>
using namespace std;
int main()
{
    int n,t,min,max;
    long long sum=0;
    cin>>t;
    max=min=t;
    for(int i=2; i<=n; i++) {
        cin>>t;
        if(t>max) max=t;
        if(t<min) min=t;
        sum=sum+t;
    }
    cout<<max<<" "<<min<<" "<<sum;
    return 0;
}
```

3.6.8 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。

输入格式

第一行包含一个整数 n。

第二行包含 n 个非负整数，为给定的数列，数列中的每个数都不大于 10000。

第三行包含一个整数 a，为待查找的数。

输出格式

如果 a 在数列中出现了，输出它第一次出现的位置(位置从 1 开始编号)，否则输出-1。

样例输入

6

1 9 4 8 3 9

9

样例输出

2

数据规模与约定

1 <= n <= 1000。

```
#include <iostream>
```

```
using namespace std;
int main()
{
    int n, a, ans;
    cin >> n;
    int s[n+1];
    for (int i = 1; i <= n; i++)
    {
        cin >> s[i];
    }
    cin >> a;
    ans = -1;
    for (int i = 1; i <= n; i++) {
        if (s[i] == a)
        {
            ans = i;
            break;
        }
    }
    cout << ans << endl;
    return 0;
}
```

3.6.9 杨辉三角形

杨辉三角形又称 Pascal 帕斯卡三角形，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。

输入格式

输入包含一个数 n。

输出格式

输出杨辉三角形的前 n 行。每一行从这一行的第一个数开始依次输出，中间使用一个空格分隔。请不要在前面输出多余的空格。

样例输入

4

样例输出

1

1 1

1 2 1

1 3 3 1

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

int main()
{
    int n,i,j;
    cin>>n;
    int a[n][n];
    for(i=0; i<=n-1; i++)
    {
        a[i][0]=1;
        a[i][i]=1;
    }

    for(i=2; i<=n-1; i++)
    {
        for(j=1; j<=i-1; j++)
        {
            a[i][j]=a[i-1][j-1]+a[i-1][j];//每个数是上面两数之和
        }
        cout<<endl;
    }

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

```
            printf("%6d",a[i][j]);
        }
        cout<<endl;
    }
    return 0;
}
```

3.6.10特殊的数字　水仙花数

153 是一个非常特殊的数，它等于它的每位数字的立方和，即153=1*1*1+5*5*5+3*3*3。编程求所有满足这种条件的三位十进制数。

输出格式

按从小到大的顺序输出满足条件的三位十进制数，每个数占一行。

方法一、

```
#include <iostream>
using namespace std;
int main()
{  int n, a, b, c;
     for(a=1;a<=9;a++)
        for (b=0;b<=9;b++)
           for(c=0;c<=9;c++)
           { n=100*a+10*b+c;
              if(a*a*a+b*b*b+c*c*c==n)
                   cout<<n<<endl;
            }
   return 0;
}
```

方法二、

```
#include <iostream>
using namespace std;
int main() {
    int i,j,k,n;
    for(n=100; n<1000; n++) {
        i=n/100;
        j=n/10%10;
        k=n%10;
        if(i*100+j*10+k==i*i*i+j*j*j+k*k*k)
            cout<<n<<endl;
```

```
    }
    return 0;
}
```

3.6.11 回文数

1221 是一个非常特殊的数，它从左边读和从右边读是一样的，编程求所有这样的四位十进制数。

输出格式

1.按从小到大的顺序输出满足条件的四位十进制数。

```
#include <iostream>
using namespace std;
int main()
{
    int a,b,c,d,i;
    for(i=1000; i<=9999; i++) {
        a=i/1000;
        b=i/100%10;
        c=i/10%10;
        d=i%10;
        if(a==d&&b==c)
            cout<<i<<endl;
    }
    return 0;
}
```

3.6.12特殊回文数

123321 是一个非常特殊的数，它从左边读和从右边读是一样的。

输入一个正整数 n， 编程求所有这样的五位和六位十进制数，满足各位数字之和等于 n 。

输入格式

输入一行，包含一个正整数 n。

输出格式

按从小到大的顺序输出满足条件的整数，每个整数占一行。

样例输入

52

样例输出

899998

989989

998899

数据规模和约定

1<=n<=54。

```
#include<iostream>
#include<cstdio>
using namespace std;
int main()
{
    int n,i,t,k,sum=0,sum2=0;
    cin>>n;
    for(i=10000; i<=999999; i++) {
        t=i;
        if(i<100000) k=10000;
        else k=100000;
        sum=0;
        sum2=0;
        while(t!=0) {
            sum=sum+t%10*k;
            sum2=sum2+t%10;
            t=t/10;
            k=k/10;
        }
        if(sum==i && sum2==n)
            cout<<i<<endl;
    }
    return 0;
}
```

方法 2

```
#include<cstdio>
using namespace std;
int main()
{
    int n,i,t,k,sum=0,sum2=0;
    cin>>n;
    for(i=10000; i<=99999; i++)
    {
```

```
        t=i;
        sum=0;  sum2=0; k=10000;
        while(t!=0)
        {
            sum=sum+t%10*k;
            sum2=sum2+t%10;
            t=t/10;
            k=k/10;
        }
        if(sum==i && sum2==n)
            cout<<i<<endl;
    }

    for(i=100000; i<=999999; i++)
    {
        t=i;
        sum=0;  sum2=0;  k=100000;
        while(t!=0)
        {
            sum=sum+t%10*k;
            sum2=sum2+t%10;
            t=t/10;
            k=k/10;
        }
        if(sum==i && sum2==n)
            cout<<i<<endl;
    }
    return 0;
}
```

3.6.13十进制转十六进制

十六进制数是在程序设计时经常要使用到的一种整数的表示方式。它有0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F 共 16 个符号，分别表示十进制数的 0 至 15。十六进制的计数方法是满 16 进 1，所以十进制数 16 在十六进制中是 10，而十进制的 17 在十六进制中是 11，以此类推，十进制的 30 在十六进制中是 1E。　　给出一个非负整数，将它表示成十六进制的形式。

输入格式

输入包含一个非负整数 a，表示要转换的数。0<=a<=2147483647

输出格式

输出这个整数的 16 进制表示

样例输入

30

样例输出

1E

```
#include<iostream>
#include<iostream>
#include<cstdio>
using namespace std;
int main()
{
    int m,n,i=0;
    int a[100];
    char                                                    hex[16]=
{'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
    cin>>n;
    while(n>0)
    {
        a[i]=n%16;
        n=n/16;
        i++;
    }
    for(i=i-1; i>=0; i--)
    {
        m=a[i];
        printf("%c",hex[m]);
    }
    return 0;
}
```

3.6.14十六进制转十进制

从键盘输入一个不超过 8 位的正的十六进制数字符串，将它转换为正的十进制数后输出。　　注：十六进制数中的 10~15 分别用大写的英文字母 A、B、C、D、E、F 表示。

样例输入

FFFF

样例输出

65535

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cmath>
int main()
{
    double sum=0,x;
    char a[8];
    int len,i=0;
    gets(a);
    len=strlen(a);
    while(len) {
        if(a[len-1]>='A'&&a[len-1]<='F')
            x=(a[len-1]-'7')*pow(16,i++);
        else
            x=(a[len-1]-'0')*pow(16,i++);
        sum+=x;
        len--;
    }
    printf("%.0lf",sum);
    return 0;
}
```

3.6.15十六进制转八进制

给定 n 个十六进制正整数，输出它们对应的八进制数。

输入格式

输入的第一行为一个正整数 n （1<=n<=10）。 接下来 n 行，每行一个由 0~9、大写字母 A~F 组成的字符串，表示要转换的十六进制正整数，每个十六进制数长度不超过 100000。

输出格式

输出 n 行，每行为输入对应的八进制正整数。

注意

输入的十六进制数不会有前导 0，比如 012A。 输出的八进制数也不能有前导 0。

样例输入

239123ABC

样例输出

714435274

提示

先将十六进制数转换成某进制数，再由某进制数转换成八进制。

//十六进制转八进制

```
#include <iostream>
#include <string>
using namespace std;
string H2O(string str) {
    string b="", o="";
    //十六进制转二进制
    for (int i=0;i<str.length();i++) {
        switch (str[i]) {
            case '0':
                b=b+"0000";
                break;
            case '1':
                b=b+"0001";
                break;
            case '2':
                b=b+"0010";
                break;
            case '3':
                b=b+"0011";
                break;
            case '4':
                b=b+"0100";
                break;
            case '5':
                b=b+"0101";
                break;
            case '6':
                b=b+"0110";
                break;
            case '7':
```

```
            b=b+"0111";
            break;
        case '8':
            b=b+"1000";
            break;
        case '9':
            b=b+"1001";
            break;
        case 'A':
            b=b+"1010";
            break;
        case 'B':
            b=b+"1011";
            break;
        case 'C':
            b=b+"1100";
            break;
        case 'D':
            b=b+"1101";
            break;
        case 'E':
            b=b+"1110";
            break;
        case 'F':
            b=b+"1111";
            break;
    }
}
if (b.length()%3==1) b="00"+b;//补0
if (b.length()%3==2) b="0"+b;
if (b.substr(0,3)=="000") b=b.substr(3);//如果前三个数是0，截掉
//二进制转八进制
for (int i=0; i<b.length()-2; i=i+3) { //取出每三位
    o=o+char((b[i]-'0')*4+(b[i+1]-'0')*2+(b[i+2]-'0')-'0');
}
return o;
```

```
}
int main() {
    int n;
    cin>>n;
    string str;
    for (int i=1;i<n;i++) {
        cin>>str;
        cout<<H2O(str)<<endl;
    }
    return 0;
}
```

3.6.16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

输入格式

第一行为一个整数 n。

第二行包含 n 个整数，为待排序的数，每个整数的绝对值小于 10000。

输出格式

输出一行，按从小到大的顺序输出排序后的数列。

样例输入

5

8 3 6 4 9

样例输出

3 4 6 8 9

```
#include<iostream>
#include<algorithm>
using namespace std;
bool cmp(int a,int b)
{
    return a>b;
}
int main()
{
    int i,n;
    cin>>n;
    int a[n];
```

```
    for (i = 0; i < n; i++)
    cin>>a[i];
    //sort(a,a+n,cmp); //降序
    sort(a,a+n); //升序
    for (i = 0; i <n; i++)
    cout<<a[i]<<" ";
}
```

3.6.17最大最小公倍数

已知一个正整数 N，问从 1- （N-1）中任选出三个数，他们的最小公倍数最大可以为多少。

输入格式

输入一个正整数 N。

输出格式

输出一个整数，表示你找到的最小公倍数。

样例输入

9

样例输出

504

数据规模与约定

1 <= N <= 106。

```
#include<iostream>
using namespace std;
int main()
    {
        long long n,ans;
        cin>>n;
        if(n%2==1)
            ans=n*(n-1)*(n-2);
        else {
            if(n%3==0)
                ans=(n-1)*(n-2)*(n-3);
            else
                ans=n*(n-1)*(n-3);
        }
        cout<<ans;
        return 0;
```

```
}
```

4 蓝桥杯白皮书 C++ 编程题模拟题

4.1 ***白皮书初级组

4.1.1 做统计

输入 10 个正整数，以空格分隔。依次输出其中的最大值、最小值以及平均值，以逗号分隔。

样例输入：

10

1 2 3 4 5 6 7 8 9 10

样例输出：

10,1,5.5

方法一：打擂法

```
#include<iostream>
using namespace std;
int main()
{
    int i,min,max,sum=0,temp,n;
    cin>>n;
    cin>>temp;
    min=max=temp;
    for(i=2;i<=n;i++)
    {
        cin>>temp;
        if(temp<min) min=temp;
        if(temp>max) max=temp;
        sum=sum+temp;
    }
    cout<<max<<","<<min<<","<<sum*1.0/n;
    return  0;
}
```

方法二：数组排序

```
#include <iostream>
```

```
#include <algorithm>
#include <cstdio>
using namespace std;
int i,n;
long long sum;
int main()
{
    cin>>n;
    int a[n];
    for(i=0;i<n;i++)
    {
        cin>>a[i];
        sum=sum+a[i];
    }
    sort(a,a+n);
    cout<<a[n-1]<<",,"<<a[0]<<",,";
    printf("%0.2f\n",sum*1.0/n);
    return 0;
}
```

4.1.2 比大小

输入一个正方形的边长（a）及一个长方形的长与宽（b、c），然后比较两个图形的面积。如果长方形面积大， 输出“RECTANGLE”；如果正方形面积大，输出“SQUARE”；如果一样大，输出“SAME”。

输入描述：

输入正整数 a、b、c（0 < a、b、c ≤ 1000），分别代表正方形的边长和长方形的长与宽。

输出描述：

如果长方形面积大，输出“RECTANGLE”；

如果正方形面积大，输出“SQUARE”；

如果一样大， 输出 “SAME”。

样例输入：

5 4 6

样例输出：

SQUARE

45

```
#include <iostream>
```

```
using namespace std;
int main()
{
    int a, b, c, s1, s2;
    cin >> a >> b >> c;
    s1 = a * a;
    s2 = b * c;
    if (s2 > s1) cout << "RECTANGLE";
    if (s1 > s2) cout << "SQUARE" ;
    if (s1 == s2) cout << "SAME" ;
    return 0;
}
```

4.1.3 数单词

编程统计输入句子中出现“lanqiao”字样的个数。（注意，“lanqiao”字样可以是不同大小写字母的组合，例如：LanQIAO、LanQiao 等）

输入描述：

输入一个字符串（0≤ 字符串长度 ≤100）。

输出描述：

输出该字符串中“lanqiao”字样出现的次数。（注意：“lanqiao”字样可以是不同大小写字母的组合）

样例输入：

Lanqiaoqingshao,lanqiaojingsai,Lanqiaoceping.

样例输出：

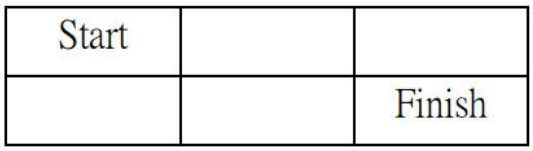
3

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str,t="lanqiao";
    cin >> str;
    int len = str.size();
    for (int i = 0; i < len; i++)
    {
        str[i] = (char)tolower(str[i]); // 转换成小写字母
    }
```

```
    cout<<str<<endl;
    int cnt = 0;
    for (int i = 0; i <= len - 7; i++)
    {
        if (t == str.substr(i, 7))
            cnt++;
    }
    cout << cnt;
    return 0;
}
```

4.1.4 查找路径

有一张 m×n 个小方格的地图，一个机器人位于地图的左上角（如图标记为 Start 的地方），它每步只能向右或者向下移动一格，如果走到右下角的终点（如图标记为 Finish 的地方），有多少种不同的方法？



例如，一个 3×2 的地图，行走的方法数是 3 种，分别是：

右 -> 右 -> 下

右 -> 下 -> 右

下 -> 右 -> 右

输入描述：

两个整数 m ($m \le 100$) 和 n ($n \le 100$)，代表地图的行数和列数。

输出描述：

一个整数，表示行走的方法数。

样例输入：

8 8

样例输出：

3432

```
#include <iostream>
using namespace std;
const int N = 100;
long long roads[N + 2][N + 2] = {0};
// roads[i][j]: 从起点 (0, 0) 到 (i, j) 的路线总数
int main() {
```

```
    int m, n;
    cin >> m >> n;
    // 第 0 行、第 0 列的格子都只有一条路线
    for (int i = 0; i < m; i++)
        roads[i][0] = 1;
    for (int i = 0; i < n; i++)
        roads[0][i] = 1;
    for (int i = 1; i < m; i++)
        for (int j = 1; j < n; j++)
            roads[i][j] = roads[i - 1][j] + roads[i][j - 1];
    cout << roads[m - 1][n - 1] << endl;
    return 0; }
```

4.2 ***白皮书中级组

4.2.1 拉线开关

小蓝家的灯是拉线式开关的，拉一次灯开，再拉一次灯关，未拉之前灯是熄灭状态。

输入一个正整数 M（1＜M＜100），作为小蓝拉动开关的次数，判断拉动 M 次后，灯是点亮状态还是熄灭状态。

输入描述：

输入一个正整数 M 作为拉动开关的次数（1＜M＜100）

输出描述：

如果灯是点亮状态输出整数“1”，如果灯是熄灭状态输出整数“0”。

样例输入：

5

样例输出：

1

```
#include <iostream>
using namespace std;
int main()
{
    int m;
    cin >> m;
    cout << m % 2 ;
}
```

4.2.2 数字组合

用户输入一个正整数 N（3<=N<=9）。从 0 到 N 之间的所有正整数（包含 0 和 N）中选择三个，组成一个三位数（0 不能作为百位数），且这个三位数为奇数，请计算出共有多少种满足条件的三位数组合。（注意：组成的三位数各位上的数字不能重复）

输入描述：输入一个正整数 N（3<=N<=9）

输出描述：输出满足条件的三位数组合的个数

样例输入：

3

样例输出：

8

上述输入输出样例的进一步解释：

用户输入的正整数，即样例输入为 3，也就是将 0、1、2、3 四个数字进行组合。符合要求的三位数为：103、123、203、213、201、231、301、321 共 8 个，所以样例输出为 8。

```
#include <iostream>
using namespace std;
int main() {
    int n, cnt = 0;
    cin >> n;
    for (int i = 1; i <= n; ++i) // 百位
        for (int j = 0; j <= n; ++j)  // 十位
                for (int k = 1; k <= n; k += 2) // 个位只能是奇数
                    if (k != i && k != j&&i != j)
                        cnt++;

    cout << cnt;
    return 0;
}
```

4.2.3 报数模拟

有 n 个人围成一个圈，从 1 到 n 按顺序排好号。然后从第一个人开始顺时针报数（从 1 到 3 报数），报到 3 的人退出圈子后，后面的人继续从 1 到 3 报数，直到留下最后一个人游戏结束，问最后留下的是原来第几号。

输入描述：

输入一个正整数 n（4 < n < 600）

输出描述：

输出最后留下的人，原来的编号是多少？

样例输入：

5

样例输出：

4

```
#include<iostream>
using namespace std;
int main()
{
    int n,m,i=1,k=0,j;   //K 用来表示报数的数
    cin>>n;
    int a[n+1]={0};    //对数据进行初始化操作
    //通过循环不断淘汰人，直到最后只剩下一个人
    while(m < n-1) //淘汰的人数小于 n-1 即不是剩余一个人
     {
        if(a[i]!=1)   k++;        //判断该人是否已经淘汰
        if(k == 3)
        {
            a[i]=1;            //淘汰报数为 3 的人（即将其值设置为 1）
            k=0;      //将 k 置为 0，重新报数
            m++;      //对淘汰人数进行计数
         }
        i++;                //对下一个人进行判断
        if(i > n) i=1;  //到了圈子的末尾出处，将下一个人置为 1
       }
     for(j=1;j<=n;j++)
          if(a[j]!=1)
         { cout<<j; break; }//只有一个 找到即退出
    return 0;
}
```

4.2.4 算天数

用户输入未来的某一天，输入格式为如 2021 6 1，编程计算这一天和今天相差多少天？（例如：今天和明天是相差一天）星期几？（注意：输出格式为 星期的数字值并在其前加“*”)。

样例输入：

2021 6 1

样例输出：

183

*2

闰年分为普通闰年和世纪闰年。

普通闰年：公历年份是 4 的倍数，且不是 100 的倍数，为普通闰年。（如 2004 年就是普通闰年）。

世纪闰年：公历年份是整百数的，必须是 400 的倍数才是世纪闰年（如 1900 年不是世纪闰年，2000 年是世纪闰年）。

总结为：四年一闰，百年不闰，四百年再闰。

闰年的一年为 366 天,闰年的二月份为 29 天。平年一年为 365 天,平年的二 月为 28 天。

每年的 1、3、5、7、8、10、12 月份为 31 天，4、6、9、11 月份为 30 天。

```
#include <iostream>
using namespace std;
bool isLeap(int y)
{
    return y % 4 == 0 && y % 100 != 0 || y % 400 == 0;
}
int main()
{
    int Y_now = 2021, M_now = 1, D_now = 14, W_now = 4; // 今日日期
    int days[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    days[2] = 28 + (int)isLeap(Y_now);
    int Y_future, M_future, D_future;
    cin >> Y_future >> M_future >> D_future;
    int y = Y_now, m = M_now, d = D_now, w = W_now;
    int cnt = 0;
    while (!(y == Y_future && m == M_future && d == D_future))
    {
        cnt++;
        w++;
        cout<<y<<'*'<<m<<'*'<<d<<endl;
        cout<<"cnt=="<<cnt<<endl;
        if (w > 7)
        {
```

```
            w = 1;

        }
        d++;

        if (d > days[m])
        {
            d = 1;
            m++;
            if (m > 12)
            {
                m = 1;
                y++;
                days[2] = 28 + (int)isLeap(y);
            }
        }
    }
    cout << cnt << endl;
    cout << '*' << w ;
    return 0;
}
```

4.2.5 标记门牌号

一家酒店有 F 层高（0＜F＜100），每层都有 n 个房间（0＜n＜100），房间门牌号由不少于 3 位的数字组成：后两位是房间号，从 1 开始，不间断地排到 n，不足两位的前面补零；前面一或两位是楼层号，从 1 开始，不间断地排到 F，前面不补零。如 1 楼第 8 个房间门牌号是 108，12 楼第 16 个房间门牌号是 1216。

现在要为每个房间制作一个门牌号码的金属牌，每个金属牌都要定制模具，数字居中显示。但如果某房间门牌上下颠倒过来的号码与原号码一模一样，就需要做一个特殊记号，以免混淆方向。

例如：8008、1691、6119、818、619 等等。

因为数字 6 倒过来是 9；9 倒过来是 6；0、1、8 倒过来还是原数；其他数字倒过来不构成数字。对于多位数 618，倒过来看应该是 819，与原来不一样， 就不用做记号了。

输入楼层数 F 和房间数 n，计算有多少房间的门牌号码需要做特殊记号。

输入描述：

输入两个正整数 F（0＜F＜100）和 n（0＜n＜100）中间一个空格隔开，代表酒店的楼层数和每层房间数。

输出描述：

输出需要做特殊记号的门牌数。

样例输入：

2 5

样例输出：

1

```
#include <iostream>
using namespace std;
bool check(int num) {
    int t[10] = {0, 1, -1, -1, -1, -1, 9, -1, 8, 6};
     // i 与 t[i] 互为颠倒
    int a = num / 1000; // 千位
    int b = num / 100 % 10;
    int c = num / 10 % 10;
    int d = num % 10; // 个 位
    if ((a == 0 && t[b] == d && t[c] == c) || (a != 0 && t[a] == d && t[b] ==
c))
        return true;
    else
```

```
        return false;
}
int main() {
    int f, n, cnt = 0;
    cin >> f >> n;
    for (int i = 1; i <= f; ++i)
     {
        for (int j = 1; j <= n; ++j)
          {
            int num = i * 100 + j;
            if (check(num))   cnt++;
            }
    }
    cout << cnt ;
    return 0; }
```

4.3***第十届蓝桥青少年创意编程省赛 C++高级组

4.3.1 水下探测器

水下探测器可以潜入湖中在任意水深进行科学探索。

湖水的最大深度为 h 米,即它在湖底时到水面的距离，0<=h<=100；

探测器最初的水下深度为 s 米，0<=s<=100；

当探测器不在水面（当前深度大于 0）时，每个 u 指令可使它上浮 1 米，而当探测器在水面时，u 指令是无效的；

当探测器不在湖底（当前深度小于 h）时，每个 d 指令可使它下沉 1 米，而当探测器在湖底时，d 指令是无效的；

在执行到无效指令时，探测器不做任何操作而继续执行下一指令。

编程实现：

根据给定的 h、s 和一个指令序列(由字符 u、d 组成的字符串，长度不超过 100)，求出执行完整的指令序列后，探测器的水下深度。

输入：

第一行：h 和 s，以空格分开。0<=s<=h<=100

第二行：长度不超过 100 的指令字符串，串中仅包含字母 u 或 d

输出：

代表探测器在执行指令后的水下深度的数字。

样例输入：

9 1

uduudd

样例输出：

2

考察知识:

基础语法,字符串,循环,条件判断

```
#include <iostream>
using namespace std;
int main(){
    int h,s;
    string str;
    cin>>h>>s>>str;
    for(int i=0;i<str.size();i++){
```

```
        if(str[i]=='u'&&s>0) s--;
        if(str[i]=='d'&&s<h) s++;
    }
    cout<<s;
}
```

4.3.2 小猫吃鱼

明明家从 1 号站点出发，开车去旅游，一共要经过 n 个站点，依次为 2、3……n。

由于明明带上了心爱的小猫，在每个站点都要为小猫提供一条鱼用做美餐（包括 1 号站点）。

除了 1 号站点只能吃 1 号站点买的鱼，其他站点既可以吃当地买的鱼，也可吃之前经过的站点买了存入车载冰箱中的鱼。但车载冰箱消耗的电能来自汽油，所以每条鱼用冰箱保存到下一站的费用与各个站点的汽油价格有关。

为使问题简化，我们约定：

（1）车从某站开出时油箱中都是此站点刚加的汽油。

（2）车载冰箱能容纳一路上需要的所有鱼。 即：每条鱼的费用既包括购买时的费用，也包括用冰箱保存鱼的费用。

编程实现：

为了降低小猫吃鱼的总代价，明明预先上网查到了这 n 个站点的鱼价和汽油价格。并据此算出每个站点买一条鱼的费用以及从该站点到下一站用冰箱保存一条鱼的费用。你能帮明明算出这一路上小猫吃鱼的最小总费用吗？

输入：

第一行：站点数 n，1<n<100。

接下来的 n 行：每行两个以空格分隔的正整数，表示：这一站买一条鱼的费用，以及从这一站把每条

鱼保存到下一站的费用，两个费用均为小于 10000 的正整数。

输出：

最小总费用，是一个正整数。

样例输入：

5

6 3

7 1

3 2

8 3

9 5

样例输出：

29

使用 for 循环，每次看看价钱是不是最划算，不是就加上一条鱼，是也要加上这一条鱼，每次还要加上保存的价钱。

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cin>>n;
    int m1[n], m2[n];
    for(int i = 0; i < n; i++)
        cin>>m1[i]>>m2[i];
    //输入当前买鱼的钱和保存的钱
    int min = 9999999, t = 0;
    for(int i = 0; i < n; i++) {
        if(min > m1[i])
            min = m1[i];
        t = t + min; //上一条鱼
        min = min + m2[i]; //保存
    }
    cout<<t<<endl;
    return 0;
}
```

4.3.3 评选最佳品牌

n 个评委投票，在 m 个商品中评选一个最佳品牌。

评选采用多轮淘汰制，即：每轮投票，淘汰掉得票最少的候选品牌（得票并列最少的品牌一起淘汰）。

如此一轮轮淘汰下去，如果最后只剩下一个品牌当选，即告评选成功。

但如果在某轮投票中，当时未被淘汰的所有候选品牌（大于等于两个品牌）都并列得票最少，即告评选失败。

如果评选成功就输出当选品牌号。否则输出最后一轮评选时唯一选票数的相反数。

在评选流程中，每个评委的态度都可用一个序列来表示；例如当 m=5 时，某评委的评选态度序列为：3、

5、1、2、4，则表示该评委：优先投 3 号，当 3 号被淘汰时投 5 号，当 3 和 5 都被淘汰时投 1，当 3、5、1 都被淘汰时投 2，仅剩 4 号时才投 4 号品牌的票。

选票的序列中可以表示弃权，用 0 来表示，例如当 m=5 时，某评委的评选态度序列为：3、5、0，则表示该评委：优先投 3 号，当 3 号被淘汰时投 5 号，其它情

况下不投任何品牌的票。

编程实现：

请你编一个程序，模拟各轮投票的过程，得到评选结果。

输入：

第一行：m(0<m<10,表示参加评选的品牌数)和 N(1<n<1000,表示参加投票的评委数)，之间以空格分隔接下来的 n 行：每行都是长度不超 m 的数字字符串，每个字符串表示一个评委的评选态度。

输出：

评选结果。

样例 1 输入：

3 4

123

213

132

10

样例 1 输出：

1

样例 2 输入：

3 4

321

213

231

312

样例 2 输出：

-2

样例数据分析:

样例 1:

第一行 3 4 代表 3 个品牌,4 个评委

第一轮投票,3 个评委优先选择 1 号品牌,1 个评委选择 2 号品牌,品牌 3 得票最少,淘汰掉

第二轮投票,3 个评委优先选择 1 号品牌,1 个评委选择 2 号品牌,品牌 2 得票最少,淘汰掉,淘汰 2 号品牌后,只剩一个 1 号品牌,1 号品牌胜出，最终结果 1

样例 2:

第一行 3 4 代表 3 个品牌,4 个评委

第一轮投票,2 个评委选择 2 号品牌,两个评委选择 3 号品牌,1 号得票最少,淘汰掉

第二轮投票,2 个评委选择 2 号品牌,两个评委选择 3 号品牌,由于只剩下两个品牌,

且并列最少,都是 2 票,代表评选失败,需要输出最后一轮票数 2 的相反数-2

最终结果 -2

考察知识:

字符串,桶排序,模拟算法

```
#include <iostream>
using namespace std;
int main(){
    int m,n;
    cin>>m>>n;
    string str[n+1];
    for(int i=1;i<=n;i++){
        cin>>str[i];
    }
    int a[m+1];//记录投票过程,-1 代表淘汰
    while(true){
        for(int i=1;i<=m;i++){
            //重新计票,所有没有被淘汰的票数归零
            if(a[i]!=-1)a[i]=0;
        }
        for(int i=1;i<=n;i++){
            //计票
            string s=str[i];
            for(int j=0;j<s.size();j++){
                char t=s[j]-'0';
                if(t==0){
                    //如果为 0 说明弃权
                    break;
                }else{
                    //不为 0,则代表要给 t 投票,如果 t 没有被淘汰则给它投票
                    if(a[t]>=0){
                        a[t]=a[t]+1;
                        break;
                    }
                }
            }
        }
```

```
        int min=m+1;
        int max=0;
        for(int i=1;i<=m;i++){
            if(a[i]>=0){
                if(min>a[i]) min=a[i];
                if(max<a[i])max=a[i];
            }
        }
        if(max>min){
            //最大值比最小值大,需要将所有最小值的淘汰
            for(int i=1;i<=m;i++){
                if(a[i]==min)a[i]=-1;
            }
        }else{
            //最大值和最小值相等说明平票
            int count=0;
            int dx=0;
            for(int i=1;i<=m;i++){
                if(a[i]==max){
                    count++;
                    dx=i;
                }
            }
            if(count>1){
                //如果数量多,说明评选失败
                cout<<0-max;
            }else{
                cout<<dx;
            }
            break;
        }
    }
    cin>>m;
}
```

4.3.4 蓝桥杯赛迷宫

把一个 n 行 m 列的字符阵列看做一个迷宫，迷宫仅包含 L、Q、B、S 中的大写字母（蓝桥杯赛的汉语拼音首字母）。

初始时，你可以从任意一个“L”字母开始，移向相邻的“Q”字母，然后从此“Q”字母出发，移向相邻的“B”字母，然后从此“B”字母出发，移向相邻的“S”字母……。这样，你就算是走过了一个“LQBS”字符序列。

接下来，仍然可以从此“S”字母出发，移向相邻的“L”字母……，重复上述的动作，你就可以不断地走过“LQBS”序列。

请注意，所谓相邻仅包含上、下、左、右 4 个方向，且只能从 L->Q，从 Q->B，从 B->S，从 S->L。

可以想像，由于选择的出发点不同，我们有可能在迷宫中走过无数次的“LQBS”，或者是有限次的“LQBS”，

或者一次也走不了。

编程实现：

请你编写程序，求出在给定的迷宫中，我们最多可以走过多少次“LQBS”?

输入：

第一行：正整数 n,m，表示迷宫的规模为 n 行 m 列，0<m<100，0<n<100

接下来的 n 行：每行 m 个符合题意的字母，字母间无空格。

输出：

一个整数。即：如果在迷宫中可以无限次的走过“LQBS”，输出-1，否则，输出可以走过“LQBS”的最多次数。

样例 1 输入：

1 2

LQ

样例 1 输出：

0

样例 2 输入：

3 3

LSB

QBQ

BSL

样例 2 输出：

-1

样例 3 输入：

4 4

BLQB
BBQS
SBQL
QQQQ
样例 3 输出：
2
样例数据分析:
样例 1:
第一行数据 3 3 代表是 3 行 3 列的迷宫
第 1 步,寻找 L
LSB
QBQ
BSL
第 2 步,在 L 的上下左右寻找 Q
LSB
QBQ
BSL
第 3 步,在 Q 的上下左右寻找 B
LSB
QBQ
BSL
第 4 步,在 B 的上下左右寻找 S
LSB
QBQ
BSL
如此当重复到 10 次的时候,仍可以继续走,代表进入无限循环
因为 3 行 3 列最多 9 个格子,能走 10 步就代表一定进入无限循环
输出-1
样例 2:
第一行数据 4 4 代表是 4 行 4 列的迷宫
第 1 步,寻找起点 L
BLQB
BBQS
SBQL
QQQQ
第 2 步,在 L 的上下左右寻找 Q

BLQB
BBQS
SBQL
QQQQ
第 3 步,在 Q 的上下左右寻找 B
BLQB
BBQS
SBQL
QQQQ
第 4 步,在 B 的上下左右寻找 S
BLQB
BBQS
SBQL
QQQQ
第 5 步,在 S 的上下左右寻找 L
BLQB
BBQS
SBQL
QQQQ
第 6 步,在 L 的上下左右寻找 Q
BLQB
BBQS
SBQL
QQQQ
第 7 步,在 Q 的上下左右寻找 B
BLQB
BBQS
SBQL
QQQQ
第 8 步,在 B 的上下左右寻找 S
BLQB
BBQS
SBQL
QQQQ
第 9 步,在 S 的上下左右寻找 L
因为找不到 L 了,则代表迷宫走到尽头,一共走了 8 步,LQBS 一共 4 个字符,所以走

了

8/4=2 共计 2 次 LQBS,输出 2

考察知识:

搜索与回朔

未学到搜索与回朔算法,可以采用循环遍历类似穷举算法进行题解

```
#include<iostream>
using namespace std;
int main(){
    int n,m;
    cin>>n>>m;
    char map[n+1][m+1];
    int path[n+1][m+1];
    for(int i=1;i<=n;i++){
        string s;
        cin>>s;
        for(int j=1;j<=m;j++){
            map[i][j]=s[j-1];
            path[i][j]=0;
            if(map[i][j]=='L'){
                path[i][j]=1;
            }
        }
    }
    string LQBS="LQBS";

    int now=1;
    while(true){
        int flag=0;
        for(int i=1;i<=n;i++){
            for(int j=1;j<=m;j++){
                if(path[i][j]==now){
                    if(map[i][j-1]==LQBS[now%4]){
                        path[i][j-1]=now+1;
                        flag++;
                    }
                    if(map[i][j+1]==LQBS[now%4]){
```

```
                    path[i][j+1]=now+1;
                    flag++;
                }
                if(map[i-1][j]==LQBS[now%4]){
                    path[i-1][j]=now+1;
                    flag++;
                }
                if(map[i+1][j]==LQBS[now%4]){
                    path[i+1][j]=now+1;
                    flag++;
                }
            }
        }
    }
    /*for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cout<<path[i][j]<<" ";
        }
        cout<<endl;
    }
    cout<<endl;*/
    if(flag==0){
        break;
    }
    if(now>(n+1)*(m+1)){
        break;
    }
    now++;
}
if(now>(n+1)*(m+1)){
    cout<<-1;
}else{
    cout<<now/4;
}
cin>>n;
}
```

4.3.5 最大购物优惠

小惠听说超市正在打折促销，要制订一个得到最大优惠的购物计划。

小惠的体力可以提起 w 单位重量的东西，还有一个能装 V 个单位体积的购物袋，并详细了解了各打折商品的重量、体积及此商品实际优惠的金额。她想在自己体力的限度和购物袋容积限度内，尽可能多地得到购物优惠。

超市规定这些打折商品每种只能购买一件。

编程实现：

请你编写程序，制定一个购买商品的计划，求出小惠能得到的最大优惠金额和实际应购买的各商品序号。

输入：

第一行：依次为 w、v 和 n(n 为商品种类数)，所有数值均为不超过 100 的正整数 接下来的 n 行：每行有三个整数，依次为某种商品的重量、体积和让利金额，数值间以空格分开，所有数值均为不超过 100 的正整数

输出：

第一行：小惠能够得到的最大让利金额

第二行：依次为从小到大排列的商品序号，序号从 1 开始，序号间用空格分开。若第二行输出的序列

不唯一，则输出其最小字典序。

样例输入：

10 9 4

8 3 6

5 4 5

3 7 7

4 5 4

样例输出：

9

2 4

考察知识:

动态规划，二维费用的背包问题，在 01 背包问题的基础上增加一个费用维度

状态转移方程：

```
yh[i][j][k]=max(yh[i-1][j][k],yh[i-1][j-w[i]][k-v[i]]+n[i])
```

经典 01 背包问题解

```
#include <iostream>
    using namespace std;
    int main() {
        int w, v, n;
        cin >> w >> v >> n;
        int a[n + 1], b[n + 1], c[n + 1];
        int yh[n + 1][w + 1][v + 1];
        string s[n + 1][w + 1][v + 1];
        for (int i = 1; i <= n; i++)
            cin >> a[i] >> b[i] >> c[i];

        for (int i = 0; i <= n; i++) {
            for (int j = 0; j <= w; j++) {
                for (int k = 0; k <= v; k++) {
                    yh[i][j][k] = 0;
                    s[i][j][k] = "";
                }
            }
        }
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= w; j++) {
                for (int k = 1; k <= v; k++) {
                    int bn = yh[i - 1][j][k];
                    int n = 0;
                    if (j >= a[i] && k >= b[i]) { //重量和体积够,拿当前物品
                        n = yh[i - 1][j - a[i]][k - b[i]] + c[i];
                    }
                    if (n > bn) { //如果拿了比不拿价值大,就拿这个物品
                        yh[i][j][k] = n;
                        //同时记录拿了这个物品
            s[i][j][k] = s[i - 1][j - a[i]][k - b[i]] + " " + (char)(i + '0');
                    } else {
                        //否则记录当前重量和体积的最优策略是不拿
                        yh[i][j][k] = bn;
                        //同时记录不拿当前物品,保持和之前一样的物品列表
                        s[i][j][k] = s[i - 1][j][k];
```

```
                }
            }
        }
    }
    cout << yh[n][w][v] << endl;
    string str = s[n][w][v];
    cout << str.substr(1, str.size() - 1);
}
```

4.3.6 购物单

XX 大促销又来了，长长的购物单，都是有打折优惠的。

请你帮他计算一下，需要从取款机上取多少现金，才能搞定这次购物。

取款机只能提供 100 元面额的纸币。小明想尽可能少取些现金，够用就行了。

你的任务是计算出，小明最少需要取多少现金。

180.90　　88 折

10.25　　65 折

56.14　　9 折

104.65　　9 折

100.30　　88 折

297.15　　半价

26.75　　65 折

130.62　　半价

```
#include<iostream>
using namespace std;
int main()
{
    double j,z,sum=0;//原价 折扣
    for(int i=1; i<=8; i++) {
        cin>>j>>z;
        sum=sum+j*z;
    }
    sum=sum+0.9; //零钱部分舍入 精确到 1 角
    if(int(sum)%100>0)//不是 100 元的整数倍
        cout<<int(sum)/100*100+100;
    else
        cout<<int(sum);
```

```
    return 0;
}
```

4.3.7 等差素数列

2,3,5,7,11,13,….是素数序列。类似：7,37,67,97,127,157 这样完全由素数组成的等差数列，叫等差素数数列。上边的数列公差为 30，长度为 6。

2004 年，格林与华人陶哲轩合作证明了：存在任意长度的素数等差数列。

这是数论领域一项惊人的成果！有这一理论为基础，请你借助手中的计算机，满怀信心地搜索：

长度为 10 的等差素数列，其公差最小值是多少？

注意：需要提交的是一个整数，不要填写任何多余的内容和说明文字。

先用埃氏筛法，把 1~N (N 先设置一个 10000 吧，不够再加)以内的素数都筛选出来，然后再枚举 1~10000(公差，不够再加)，寻找连续 10 个的素数。

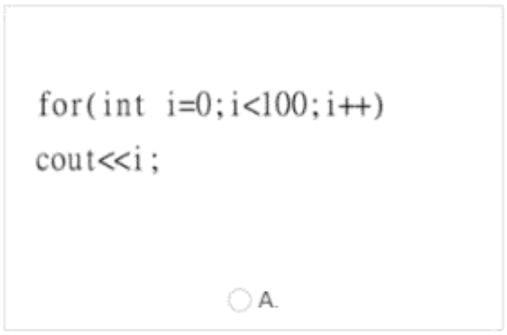
```
#include <iostream>
using namespace std;
const int maxn = 10000000;
int prime[maxn];
bool is_prime[maxn + 10];  //is_prime[i]为 true 表示 i 是素数
bool is_Prime(int n)
{
    int i = 0;
    for (i = 2; i * i <= n; i++)
    {
        if (n % i == 0) return false;
    }
    return n != 1;
}
//返回 n 以内的素数
int sieve(int n)
{
    int p = 0;
    //初始化
    for (int i = 0; i <= n; i++) {
        is_prime[i] = true;
    }
    is_prime[0] = is_prime[1] = false;
```

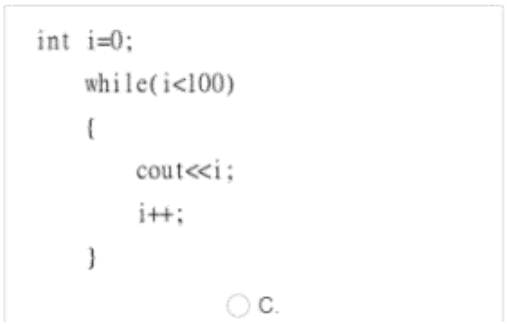
```
    for (int i = 0; i <= n; i++)
    {
        if (is_prime[i])
        {
            prime[p++] = i;  //将素数添加到 prime 中
            //1.首先 2 是素数, 然后划去所有 2 的倍数
            //2.表中剩余的最小数字是 3, 他不能被更小的数整除,所以是素数
            //再将表中所有 3 的倍数都划去
            //3.以此类推, 如果表中剩余的最小数字是 m 时,m 就是素数。然后将表
中所有 m 的倍数都划去
            for (int j = 2 * i; j <= n; j += i) {
                is_prime[j] = false;
            }
        }
    }
    return p;
}
void solve()
{
    int N = 10000;
    int cnt = sieve(N);

    //公差
    for (int d = 10; d < N; d++)
    {
        //枚举 N 以内所有素数
        for (int i = 0; i < cnt; i++)
        {
            int tmp = prime[i],
                flag = true;
            //是否连续 10 个都为素数
            for (int j = 0; j < 9; j++)
            {
                if (tmp + d > N || !is_Prime(tmp + d))
                {
                    flag = false;
```

```
                    break;
                }
                else
                {
                    tmp += d;  //下一个素数
                }
            }
            if (flag) {
                cout << d << " " << prime[i] << endl;
                return;
            }
        }
    }
}
int main()
{
    solve();
    return 0;

}
```

4.4***第十一届蓝桥青少组省赛C++高级组（20年3月）

选择题：
1.结构化程序所要求的基本结构不包括（ B ）。
A、顺序结构
B、GOTO（）跳转
C、选择（分支）结构
D、重复（循环结构）
2.若定义 int a=2，b=2，下列表达式中值不为4的是（ A ）
A.a*（++b）
B.a*（b++）
C.a+b
D.a*b
3. 在下列选项中，不能输出100个整数的是（ D ）。





4.下列叙述中正确的是（ B ）。
A. 线性表的链式存储结构与顺序存储结构所需要的存储空间是相同的
B. 线性表的链式存储结构所需要的存储空间一般要多于顺序存储结构
C. 线性表的链式存储结构所需要的存储空间一般要少于顺序存储结构
D. 上述三种说法都不对
5. 小蓝打羽毛球实行积分赛制，获胜积5分，打平积2分，失败扣1分。已知小蓝在20场积分赛后积61分且有3场比赛打平，那么小蓝的胜率为：（ C ）A. 48%
B. 55%
C. 60%
D. 75%
// 3平 12胜 5负

4.4.1 属相

我们中国人对老鼠的感情可不一般，鼠是中国传统十二生肖之首。那么 2020 年出生的“20 后”是否都是“鼠宝宝”呢？其实不是，2020 年 1 月 1 日～1 月 24 日出生的“20 后”，仍然是“猪宝宝”，因为他们出生在农历己亥猪年；大年初一（1 月 25 日）及之后出生的“20 后”才是“鼠宝宝”。那么接下来请你判断一下，以下生日的宝宝是“猪宝宝” 还是“鼠宝宝”?

输入：符合常识的两个空格分隔的整数 month,day，分别代表宝宝出生的月份及日子，（1<=month<=12, 1<=day<=31）。

输出：若是“猪宝宝”请输出"Pig"；若是“鼠宝宝”请输出"Mouse"。样例输入：

1 1

样例输出： Pig

```
#include<iostream>
using namespace std;
int main()
{
    int month,day;
    cin>>month>>day;
    if(month==1&&day<=24)
          cout<<"Pig";
       else
          cout<<"Mouse";
    return 0;
}
```

4.4.2 写个“2”

题目描述:2020 年 2 月，小蓝参加“蓝桥杯大赛青少年创意编程 C++组”选拔赛。在一个这么“2”的时间里参赛，小蓝一时高兴，忍不住在键盘上敲出了一个会写“2”的程序。

输入：一个整数 n(3<=n<=100)

输出：一个由“*”组成的长、宽都是 n 的“2”字图形,具体请参见样例。

样例输入 1： 5

```
*****
    *
   *
  *
*****
```

样例输出 1：

样例输入 2： 8

样例输出 2：********

```
#include <iostream>
using namespace std;
int main( ) {
    int n;
    cin>>n;
    for(int i=1; i<=n; i++)
    {
        for(int j=1; j<=n; j++)
        {//第一行 第 n 行    i+j 的和等于 n+1
            if(i==1||i==n||i+j==n+1)
                cout<<"*";
            else
                cout<<" ";
        }
    cout<<endl;
    }
return 0;
}
```

4.4.3 石头剪刀布

放假期间，小蓝与电脑对垒，玩起了一款经典的游戏：“石头剪刀布”。游戏规则想必大家已经非常熟悉了：两边一样则为平局，否则石头胜于剪刀；剪刀胜于布；布胜于石头。小蓝与电脑的对垒一共有 n 个回合，平局或败局得分为 0；胜局得分取决于小蓝出手的阵容， 剪刀、石头、布各有不同的分值：

出手“石头”赢的话得 r 点分值； 出手“剪刀”赢的话得 s 点分值； 出手“布”赢的话得 c 点分值；

但是，在第 i 回合中，小蓝不能使用在第（i-k）个回合中使用的阵容。（在前 k 个回合中，小蓝可以使用任何阵容。）

在游戏开始之前，电脑已经事先安排好了每回合比赛的阵容，而小蓝居然未卜先知了电脑的阵容！电脑的出手阵容用字符串 t 给出，如果 t 的第 i 个字符（1≤i≤n）为 r，则代表电脑将在第 i 个回合中出手“石头”。同样，c 和 s 分别代表“布”和“剪刀”。

那么请你计算一下，小蓝在游戏中可以获得的最大分值是多少？ 输入:

n k

r s c t

其中：n,k,r,s,c 都是整数，t 是字符串。2≤n≤20

1≤k≤n−1

1≤r,s,c≤1000

字符串 t 的长度是 n 输出:

小蓝在游戏中可以获得的最大分值。

6

样例输入:

5 2

8 7 6

rsrcr

样例输出:

27

样例说明：

机器出手的阵容是：石头、剪刀、石头、布、石头

则小蓝出手：布、石头、石头、剪刀、布，分值为 6+8+0+7+6=27 分

第 3 回合里，小蓝不能再出第（3-2=1）回合里出过的“布”了，所以选择了平局，出手“石头”，得 0 分。

```
#include <iostream>
using namespace std;
int main( ) {
    int n,k,r,s,c;
    char str1[22],str2[22];
    cin>>n>>k>>r>>s>>c>>str1;
    int sum=0;
    for(int i=0; i<k; i++) {
        if(str1[i]=='r') {
            str2[i]='c';
            sum+=c;
        } else if(str1[i]=='s') {
            str2[i]='r';
            sum+=r;
        } else if(str1[i]=='c') {
            str2[i]='s';
            sum+=s;
```

```
        }
    }
    for(int i=k; i<n; i++) {
        if(str1[i]=='r') {
            if(str2[i-k]=='c')   str2[i]='r';
            else {
                str2[i]='c';
                sum+=c;
            }
        } else if(str1[i]=='s') {
            if(str2[i-k]=='r')   str2[i]='s';
            else {
                str2[i]='r';
                sum+=r;
            }
        } else if(str1[i]=='c') {
            if(str2[i-k]=='s')   str2[i]='c';
            else {
                str2[i]='s';
                sum+=s;
            }
        }
    }
    cout<<sum;
    return 0;
}
```

4.4.4 部分排序

一个数列 P 中有 n 个数。小蓝从中选择位置连续的 k 个数，并对这 k 个数进行升序排列。求排序后的数列有多少种?

输入:

n k

P0 P1 … Pn−1

其中：所有的输入都是整数，2≤n≤100，2≤k≤n，0≤Pi≤n−1，P0,P1,…,Pn−1 数值都不相同。输出:

部分排序后数列的排列数。样例输入:

5 3

0 2 1 4 3
样例输出:
2
样例说明：从原数列抽取连续 3 个数排序后有 2 种可能性: (0,1,2,4,3) 和 (0,2,1,3,4)。

```
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;
const int N = 110;
//b 记录本次排序后的数列
//last 记录上次排序得到的数列
int a[N], b[N], last[N];
int n, k;
//检查两个数列是否相同
bool check(int a[], int b[]) {
    for(int i = 0; i < n; i ++)
        if(a[i] != b[i]) return false;
    return true;
}
int main() {
    cin >> n >> k;
    for(int i = 0; i < n; i ++) cin >> a[i];
    //将原数列 a 拷贝到 last 中
    memcpy(last, a, sizeof a);
    int sum = 0;
    //枚举所有连续 k 个数的起始位置
    for(int i = 0; i <= n - k; i ++) {
        //将 a 拷贝到 b 中进行排序
        memcpy(b, a, sizeof a);
        sort(b + i, b + i + k);
        //排序后与上一个序列不相同
        if(!check(last, b)) sum ++;
        //将 b 拷贝到 last
        memcpy(last, b, sizeof b);
    }
```

```
    cout << sum << endl;
    return 0;
}
```

4.4.5 题目的分数值

蓝桥杯 C++青少组的比赛有 n 个问题，现在请你给这 n 个问题分配分值。

n 个问题已经按从简单到困难排好序，第 i 个问题的分值是 Ai。n 个问题的分值满足如下关系：

1≤A1≤A2≤…≤An≤n。不同的问题可以具有相同的分值。

主办方希望：解决更多问题的参赛者的排名更高。 因此，对于任何解决了 k（1≤k≤n-1）个问题的参赛者，其分数总和一定要小于解决了任何 k + 1 个问题的参赛者的分数总和。

你有几种分配分值的方法？ 将答案对素数 m 取余后输出。输入：

整数 n 和 m

其中 2≤n≤5000,9×108<m<109 ，m 为素数。输出：

分值分配的方案数对 m 取余后的数字样例输入 1：

2 998244353

样例输出 1： 3

样例 1 说明：

2 个题的分值分配有 3 种方案： (1,1), (1,2), (2,2)。

样例输入 2： 3 998244353

样例输出 2： 7

样例 2 说明：

3 个题的分值分配有 7 种方案：(1,1,1), (1,2,2), (1,3,3), (2,2,2), (2,2,3), (2,3,3), (3,3,3)。

```
    //动态规划题
#include<iostream>
#include<iomanip>
using namespace std;
int dp[502][502];
int main()
{
    int n,m;
    cin>>n>>m;
    for(int k=1;k<=n;k++) dp[1][k] = k;
    for(int i=2;i<=n;i++)  // 后面还有几个数
    {
```

```
        for(int k=1;k<=n;k++)  //当前阶段有多少种选择
        {
            if(k==1) dp[i][k] = dp[i-1][k];
            else   dp[i][k] =(dp[i][k-1] + dp[i-1][k]) % m;
        }
    }
    int ans = 0;
    for(int i=1;i<=n;i++)
    {
        for(int j=i;j<=n;j++)
        {
            int k1= min(i,n-j+1);
            ans = (ans + dp[n-2][k1]) % m;
        }
    }
    if(n==2) ans = 3;
    cout<<ans<<endl;
    return 0;
}
```

4.4.6 凑算式

```
      B        DEF
A + --- + ------- = 10
      C        GHI
```

这个算式中 A~I 代表 1~9 的数字，不同的字母代表不同的数字。

比如：

6+8/3+952/714 就是一种解法，

5+3/1+972/486 是另一种解法。

这个算式一共有多少种解法？

注意：你提交应该是个整数，不要填写任何多余的内容或说明性文字。

答案：29

```
#include<stdio.h>
int ans = 0;
int num[10];
bool visit[10];
```

```
void Solve()
{
    double sum = num[0] + (double)num[1] / num[2] +
(double)(num[3]*100+num[4]*10+num[5])/(num[6]*100+num[7]*10+num[8]);
    if(sum == 10)
    {
        ans ++;
    }
}

void dfs(int index)
{
    if(index == 9)
    {
        Solve();
        return ;
    }
    for(int i = 1 ; i < 10 ; i ++)
    {
        if(!visit[i])
        {
            visit[i] = true;
            num[index] = i;
            dfs(index+1);
            visit[i] = false;
        }
    }
}

int main()
{
    dfs(0);
    printf("%d\n",ans);
    return 0;
}
```

4.4.7 快速排序

排序在各种场合经常被用到。快速排序是十分常用的高效率的算法。

其思想是：先选一个“标尺”，用它把整个队列过一遍筛子，以保证：其左边的元素都不大于它，其右边的元素都不小于它。这样，排序问题就被分割为两个子区间。

再分别对子区间排序就可以了。

下面的代码是一种实现，请分析并填写划线部分缺少的代码。

```
#include <stdio.h>
void swap(int a[], int i, int j)
{
    int t = a[i];
    a[i] = a[j];
    a[j] = t;
}
int partition(int a[], int p, int r)
{
    int i = p;
    int j = r + 1;
    int x = a[p];
    while(1){
        while(i<r && a[++i]<x);
        while(a[--j]>x);
        if(i>=j) break;
        swap(a,i,j);
    }
    ____________________;
    return j;
}
void quicksort(int a[], int p, int r)
{
    if(p<r){
        int q = partition(a,p,r);
        quicksort(a,p,q-1);
        quicksort(a,q+1,r);
    }
}
```

```
int main()
{
    int i;
    int a[] = {5,13,6,24,2,8,19,27,6,12,1,17};
    int N = 12;

    quicksort(a, 0, N-1);

    for(i=0; i<N; i++) printf("%d ", a[i]);
    printf("\n");

    return 0;
}
```

答案：swap(a,p,j)

4.4.8 抽签

X 星球要派出一个 5 人组成的观察团前往 W 星。

其中：

A 国最多可以派出 4 人。

B 国最多可以派出 2 人。

C 国最多可以派出 2 人。

....

那么最终派往 W 星的观察团会有多少种国别的不同组合呢？

下面的程序解决了这个问题。

数组 a[] 中既是每个国家可以派出的最多的名额。

程序执行结果为：

DEFFF

CEFFF

CDFFF

CDEFF

CCFFF

CCEFF

CCDFF

CCDEF

BEFFF

BDFFF

BDEFF

BCFFF

BCEFF

BCDFF

BCDEF

....

(以下省略，总共 101 行)

```
#include <stdio.h>
#define N 6
#define M 5
#define BUF 1024
void f(int a[], int k, int m, char b[])
{
    int i,j;

    if(k==N){
        b[M] = 0;
        if(m==0) printf("%s\n",b);
        return;
    }

    for(i=0; i<=a[k]; i++){
        for(j=0; j<i; j++) b[M-m+j] = k+'A';
        _____________________;  //填空位置
    }
}
int main()
{
    int  a[N] = {4,2,2,1,1,3};
    char b[BUF];
    f(a,0,M,b);
    return 0;
}
```

答案 f(a,k+1,m-j,b)或 f(a,k+1,m-i,b)

4.4.9 方格填数（DP）

如下的 10 个格子

```
   +--+--+--+
```

```
   | | | |
+--+--+--+--+
| | | | |
+--+--+--+--+
| | | |
+--+--+--+
```

填入 0~9 的数字。要求：连续的两个数字不能相邻。（左右、上下、对角都算相邻）

一共有多少种可能的填数方案？请填写表示方案数目的整数。

答案是:1580

```
#include <stdio.h>
#include <math.h>
int flag[3][4]; //表示哪些可以填数
int mpt[3][4]; //填数
bool visit[10];
int ans = 0;
void init()   //初始化
{
    int i,j;
    for(i = 0 ; i < 3 ; i ++)
        for(j = 0 ; j < 4 ; j ++)
            flag[i][j] = 1;
    flag[0][0] = 0;
    flag[2][3] = 0;
}

void Solve()
{
    int dir[8][2] = { 0,1,0,-1,1,0,-1,0,1,1,1,-1,-1,1,-1,-1};
    int book = true;
    for(int i = 0 ; i < 3 ; i ++)
    {
        for(int j = 0 ; j < 4; j ++)
        {
            //判断每个数周围是否满足
            if(flag[i][j] == 0)continue;
```

```
            for( int k = 0 ; k < 8 ; k ++)
            {
                int x,y;
            x = i + dir[k][0];
                y = j + dir[k][1];
                if(x < 0 || x >= 3 || y < 0 || y >= 4 || flag[x][y] == 0) continue;
                if(abs(mpt[x][y] - mpt[i][j]) == 1)  book = false;
            }
        }
    }
    if(book) ans ++;
}

void dfs(int index)
{
    int x,y;
    x = index / 4;
    y = index % 4;
    if( x == 3)
    {
        Solve();
        return;
    }
    if(flag[x][y])
    {
        for(int i = 0 ; i < 10 ; i ++)
        {
            if(!visit[i])
            {
                visit[i] = true;
                mpt[x][y] = i;
                dfs(index+1);
                visit[i] = false;
            }
        }
    }
```

```
    else
    {
        dfs(index+1);
    }
}
int main()
{
    init();
    dfs(0);
    printf("%d\n",ans);
    return 0;
}
```

4.4.10剪邮票

如图 1.jpg, 有 12 张连在一起的 12 生肖的邮票。

现在你要从中剪下 5 张来，要求必须是连着的。

（仅仅连接一个角不算相连）

比如，图 2.jpg，图 3.jpg 中，粉红色所示部分就是合格的剪取。

请你计算，一共有多少种不同的剪取方法。

请填写表示方案数目的整数。

注意：你提交的应该是一个整数，不要填写任何多余的内容或说明性文字。

答案：116

```
#include <stdio.h>
#include <string.h>
int mpt[3][4];
int mpt_visit[3][4];
int num[6];
int have[13];
int visit[13];
int ans = 0;
int Count = 0;

void init()
{
    int k = 1;
    for(int i = 0 ; i < 3 ; i ++)
        for(int j = 0 ; j < 4 ; j ++)
```

```
        {
            mpt[i][j] = k;
            k ++;
        }
}
int dir[4][2] = {0,1,0,-1,-1,0,1,0};
//判断五个数是否能连在一起
void dfs_find(int x,int y)
{
    for(int i = 0 ; i < 4 ; i++)
    {
        int tx,ty;
        tx = x + dir[i][0];
        ty = y + dir[i][1];
        if(tx < 0 || tx >= 3 || ty < 0 || ty >= 4) continue;
        if(have[mpt[tx][ty]] == 0 || mpt_visit[tx][ty])continue;
        mpt_visit[tx][ty] = 1;
        Count ++;
        dfs_find(tx,ty);
    }
}

void Solve()
{
    int i;
    memset(have,0,sizeof(have));
    memset(mpt_visit,0,sizeof(mpt_visit));
    for(i = 1; i < 6 ; i ++) have[num[i]] = 1;
    for(i = 0 ; i < 12 ; i ++)
    {
        int x,y;
        x = i / 4;
        y = i % 4;
        if(have[mpt[x][y]])
        {
            Count = 1;
```

```
            mpt_visit[x][y] =1;
            dfs_find(x,y);
            break;
        }
    }
    if(Count == 5)
    {
        ans ++;
    }
}

//创建 5 个数的组合
void dfs_creat(int index)
{
    if(index == 6)
    {
        Solve();
        return;
    }
    for(int i = num[index-1] + 1; i < 13 ; i ++)
    {
        if(!visit[i])
        {
            visit[i] = true;
            num[index] = i;
            dfs_creat(index+1);
            visit[i] = false;
        }
    }
}

int main()
{
    init();
    dfs_creat(1);
    printf("%d\n",ans);
```

```
    return 0;
}
```

4.4.11四平方和

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

程序输入为一个正整数 N (N<5000000)

要求输出 4 个非负整数，按从小到大排序，中间用空格分开

例如，输入：

5

则程序应该输出：

0 0 1 2

再例如，输入：

12

则程序应该输出：

0 2 2 2

再例如，输入：

773535

则程序应该输出：

1 1 267 838

请严格按要求输出，不要画蛇添足地打印类似：“请您输入...”的多余内容。

所有代码放在同一个源文件中，调试通过后，拷贝提交该源码。

注意: main 函数需要返回 0

注意: 只使用 ANSI C/ANSI C++ 标准，不要调用依赖于编译环境或操作系统的特殊函数。

注意: 所有依赖的函数必须明确地在源文件中 #include <xxx>， 不能通过工程设置而省略常用头文件。

提交时，注意选择所期望的编译器类型。

答案：

方法一：

```
#include <stdio.h>
#include <math.h>
int main()
{
    int n;
    int flag = false;
    scanf("%d",&n);
    for(int i = 0 ; i * i <= n ; i ++)
    {
        for(int j = 0 ; j * j <= n ; j ++){
            for(int k = 0 ; k * k <= n ; k ++)
            {
                int temp = n - i*i - j*j - k*k;
                double l = sqrt((double) temp);
                if(l == (int)l )
                {
                    printf("%d %d %d %d\n",i,j,k,(int)l);
                    flag = true;
                    break;
                }
            }
            if(flag)break;
        }
        if(flag)break;
    }
    return 0;
}
```

方法二：

```
#include <stdio.h>
#include <math.h>
int mpt[5000010] ={0};  //mpt[i] = 1 表示 i 能够用两个完全平方数相加而得。
int n;
void init()
{
```

```
    for(int i = 0 ; i*i <= n ; i ++)
        for(int j = 0 ; j*j <=n ; j ++)
            if(i*i+j*j <= n) mpt[i*i+j*j] = 1;
}
int main()
{

    int flag = false;
    scanf("%d",&n);
    init();
    for(int i = 0 ; i * i <= n ; i ++)
    {
        for(int j = 0 ; j * j <= n ; j ++){
            if(mpt[n - i*i - j*j] == 0) continue;   //如果剩下的差用两个完
全平方数不能组合出来就不继续
            for(int k = 0 ; k * k <= n ; k ++)
            {
                int temp = n - i*i - j*j - k*k;
                double l = sqrt((double) temp);
                if(l == (int)l )
                {
                    printf("%d %d %d %d\n",i,j,k,(int)l);
                    flag = true;
                    break;
                }
            }
            if(flag)break;
        }
        if(flag)break;
    }
    return 0;
}
```

4.4.12交换瓶子

有 N 个瓶子，编号 1～N，放在架子上。

比如有 5 个瓶子：

2 1 3 5 4

要求每次拿起 2 个瓶子，交换它们的位置。

经过若干次后，使得瓶子的序号为：

1 2 3 4 5

对于这么简单的情况，显然，至少需要交换 2 次就可以复位。

如果瓶子更多呢？你可以通过编程来解决。

输入格式为两行：

第一行: 一个正整数 N（N<10000）, 表示瓶子的数目

第二行：N 个正整数，用空格分开，表示瓶子目前的排列情况。

输出数据为一行一个正整数，表示至少交换多少次，才能完成排序。

例如，输入：

5

3 1 2 5 4

程序应该输出：

3

再例如，输入：

5

5 4 3 2 1

程序应该输出：

2

```
#include<iostream>
using namespace std;
int a[10005];//a[i]表示位置 i 的瓶子编号
int b[10005];//b[i]表示 i 号瓶子的位置。没有这个数组的话，需要在 a 数组中循环找到 i 号瓶子（O(n)），用这个数组的话是 O(1)
int main() {
    int N;
    int i;
    int id;
    int sum;
    //设瓶子 1 为位置 i 的瓶子，瓶子 2 为 i 号瓶子
    int id1, id2; //瓶子 1 的编号，瓶子 2 的编号
    int pos1, pos2; //瓶子 1 的位置，瓶子 2 的位置
    while (cin >> N) {
        for (i = 1; i <= N; ++i) {
            cin >> id;
            a[i] = id; //位置 i 放 id 号瓶子
```

```
            b[id] = i; //id 号瓶子放到位置 i
        }
        sum = 0;
        for (i = 1; i <= N; ++i) { //位置从 1 遍历到 N
            if (a[i] == i)continue; //位置 i 放的是 i 号瓶子
            //否则，瓶子 2 与瓶子 1 交换
            id1 = a[i];
            pos1 = i;
            id2 = i;
            pos2 = b[i]; //没有 b 数组的话，需要在 a 数组中找到 i 号瓶子
            ++sum;
            //瓶子 1 放到瓶子 2 的位置
            a[pos2] = id1; //瓶子 2 的位置（pos2）放瓶子 1（id1）
            b[id1] = pos2; //瓶子 1（id1）放到瓶子 2 的位置（pos2）
            //瓶子 2 放到瓶子 1 的位置
            a[pos1] = id2; //瓶子 1 的位置放瓶子 2
            b[id2] = pos1; //瓶子 2 放到瓶子 1 的位置
        }
        cout << sum << endl;
    }
    return 0;
}
```

4.4.13最大比例

X 星球的某个大奖赛设了 M 级奖励。每个级别的奖金是一个正整数。

并且，相邻的两个级别间的比例是个固定值。

也就是说：所有级别的奖金数构成了一个等比数列。比如：

16,24,36,54

其等比值为：3/2

现在，我们随机调查了一些获奖者的奖金数。

请你据此推算可能的最大的等比值。

输入格式：

第一行为数字 N (0<N<100)，表示接下的一行包含 N 个正整数

第二行 N 个正整数 Xi(Xi<1 000 000 000 000)，用空格分开。每个整数表示调查到的某人的奖金数额

要求输出：

一个形如 A/B 的分数，要求 A、B 互质。表示可能的最大比例系数
测试数据保证了输入格式正确，并且最大比例是存在的。
例如，输入：
3
1250 200 32
程序应该输出：
25/4
再例如，输入：
4
3125 32 32 200
程序应该输出：
5/2
再例如，输入：
3
549755813888 524288 2
程序应该输出：
4/1
答案：

```
#include <stdio.h>
#include <algorithm>
#include <queue>
using namespace std;
#define LL long long
struct fs
{
    LL up,down;
};
int n;
LL arr[110];
fs Fs[110];

bool cmp(LL a,LL b)
{
    return a > b;
}
```

```
LL Gcd(LL a,LL b)
{
    if( b == 0 )return a;
    return Gcd(b,a%b);
}
LL Get(LL a, LL b)
{
    if( a < b) a ^= b ^= a ^= b;
    LL v[30];
    queue<LL>team;
    if( a == b || a / b == a) return b;
    v[0] = a, v[1] = b;
    v[2] = a / b;
    int top = 3,i,j;
    team.push(a/b);
    while(team.size())
    {
        LL now = team.front();
        team.pop();
        for(i = 0 ; i < top ; i ++)
        {
            LL temp = (v[i] > now) ? v[i] / now : now / v[i];
            bool find = false;
            for(j = 0 ; j < top ; j ++)
                if( v[j] == temp) find = true;
            if(find == true) continue;
            team.push(temp);
            v[top++] = temp;
        }
    }
    LL ans = v[0];
    for(i = 0 ; i < top ; i ++)
        if(v[i] != 1)
        {
            ans = v[i];
            break;
```

```
        }
    for(i = 0 ; i < top ; i ++)
        if( v[i] < ans && v[i] != 1) ans = v[i];
    return ans;
}
int main()
{
    int i,j;
    scanf("%d",&n);
    for(i = 0 ; i < n ; i ++) scanf("%lld",&arr[i]);
    sort(arr,arr+n,cmp);
    int top = 1;
    for(i = 1; i < n ; i ++)
        if(arr[i] != arr[i-1]) arr[top++] = arr[i];
    n = top;
    for(i = 0 ; i < n - 1; i ++)
    {
        LL gcd = Gcd(arr[i],arr[i+1]);
        Fs[i].up = arr[i] / gcd;
        Fs[i].down = arr[i+1] / gcd;
    }
    LL x = Fs[0].up;
    for(i = 0 ; i < n - 1 ; i ++)
        x = Get(x,Fs[i].up);
    LL y = Fs[0].down;
    for(i = 0 ; i < n - 1; i ++)
        y = Get(y,Fs[i].down);
    printf("%lld/%lld\n",x,y);
    return 0;
}
```

4.4.14外卖店优先级

“饱了么”外卖系统中维护着 N 家外卖店，编号 1 ～ N。每家外卖店都有 一个优先级，初始时 (0 时刻) 优先级都为 0。

每经过 1 个时间单位，如果外卖店没有订单，则优先级会减少 1，最低减 到 0; 而如果外卖店有订单，则优先级不减反加，每有一单优先级加 2。

如果某家外卖店某时刻优先级大于 5，则会被系统加入优先缓存中;如果 优先级小于等于 3，则会被清除出优先缓存。

给定 T 时刻以内的 M 条订单信息，请你计算 T 时刻时有多少外卖店在优 先缓存中。

输入

第一行包含 3 个整数 N、M 和 T。

以下 M 行每行包含两个整数 ts 和 id，表示 ts 时刻编号 id 的外卖店收到一个订单。

输出

输出一个整数代表答案。

样例输入

2 6 6

1 1

5 2

3 1

6 2

2 1

6 2

样例输出

1

样例解释

6 时刻时，1 号店优先级降到 3，被移除出优先缓存；2 号店优先级升到 6， 加入优先缓存。所以是有 1 家店 (2 号) 在优先缓存中。

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n, m, t, ts, id, cnt = 0;
    int level;
    bool is_priority = false;
    vector<int> v[100005];
    cin >> n >> m >> t;
    for (int i = 1; i <= m; i++)
    {
        cin >> ts >> id;
        v[id].push_back(ts);
```

```
    }
    for (int i = 1; i <= n; i++)
    {
        if (v[i].size() == 0)
            continue;
        sort(v[i].begin(), v[i].end());
        is_priority = false;
        level = 2;
        for (int j = 1; j < v[i].size(); j++)
        {
            if (v[i][j] - v[i][j - 1] <= 1)
                level += 2;
            else
            {
                level -= (v[i][j] - v[i][j - 1] - 1);
                if (level < 0)
                    level = 0;

                if (level <= 3)
                    is_priority = false;
                level += 2;
            }
            if (level > 5)
            {
                is_priority = true;
            }
        }
        int k = v[i][v[i].size() - 1];
        level -= (t - k);
        if (level <= 3)
            is_priority = false;
        if (is_priority == true)
            cnt++;
    }
cout << cnt << endl;
     return 0; }
```

4.5 ***第十二届蓝桥杯青少组 stema 选拔赛中级组 2021 年 8 月

4.5.1 蜗牛爬井

一只蜗牛从 15 米深的井底向井口爬，白天爬 3 米，晚上向下滑落 1 米，第二天白天爬 3 米，晚上再滑落 1 米，问按照这样的规律，蜗牛多少天可以爬到井口？

```
#include<iostream>
using namespace std;
int main()
{
int n;
cin>>n;
cout<<(n-3)/(3-1)+1;
return 0;
}
```

4.5.2 判断三角形

给定三个正整数，分别表示三条线段的长度，判断这三条线段能否构成一个三角形。输入输入共一行，包含三个正整数，分别表示三条线段的长度，数与数之间以一个空格分开。 输出如果能构成三角形，则输出“yes” ，否则输出“no”。

样例输入：3 4 5

样例输出：Yes

```
#include<iostream>
using namespace std;
int main() {
    int a,b,c;
    cin>>a>>b>>c;
    if(a+b>c&&a+c>b&&b+c>a) //任意两条边之和都大于第三边
        cout<<"yes";
    else
        cout<<"no";
    return 0;
}
```

4.5.3 特殊的秒表

有两个特殊的秒表表。A 秒表一圈有 18 个刻度（指针转一圈为 18 秒），B 秒表一圈有 27 个刻度（指钟转一圈为 27 秒），A、B 秒表初始位置指针分别指向刻度 13 和 27。

同时按下两块秒表的开始按钮，两块秒表指针同时开始顺时针转动， 每秒走一个刻度，指针会持续转动。当 A 表转了 n 圈时， 那在 n 圈中 A、B 秒表指针同时指向相同刻度值的次数有多少次?

如 A 秒表转了 1 圈时 A、B 秒表同时指向相同刻度值为 1，2，3，4，5，6，7，8，9，10，11，12，13,14， 15，16，17，18。故有 18 次。

输入一个正整数 n（0<n<10^10)表示 A 秒表转动的圈数， 输出一个整数，表示在 n 圈中 A、B 秒表指针同时指向相同刻度值的次数。

```
#include<iostream>
using namespace std;
int main() {
    long long n;
    int a=18,b=27,count=1;
    cin >> n;
    for (int i = 1; i <= n*18; i++) {
        a++;
        b++;
        a=a%18;
        b=b%27;
        if(a==b) count++;
    }
    cout<<count;
return 0;
}
```

4.6***第 12 届蓝桥杯青少组 6 月省赛 c++高级组

一、选择题

1、表达式 6-1 的值是（A）。

A.整数 5　　B.字符 5　　C.表达式不合法　　D.字符 6

2、若二维数组 a 有 n 列，则在 a[i][j]前元素个数为（B）。

A.i*n+j-1　　B.i*n+j　　C.j*n+i　　D.i*n+j+1

3、以下叙述中正确的是（C）。

A.break 语句只能用于 switch 语句体中。

B.continue 语句的作用是：使程序的执行流程跳出包含它的所有循环。

C.break 语句只能用在循环体内和 switch 语句体内。

D.在循环体内使用 break 语句和 continue 语句的作用相同。

4、按照“先进后出"原则组织数据的结构是（B）。（30 分）

A.队列　　B.栈　　C.双向链表　　D.二叉树

5、用 0、1、2、3、4 这 5 个数字，能组成多少个没有重复数字的多位偶数？（）

A.144　　B.147　　C.160　　D.163

解题分析：
五位数　24 个
个位是零：4*3*2=24（依次为万位、千位、百位、十位）
四位数 60 个
个位是零：4*3*2=24（依次为千位、百位、十位）
个位不是零：2*3*3*2=36（依次为个位、千位、百位、十位）
三位数　96 个
个位是零：4*3*2
个位不是零：4*4*3*2
两位数　10 个
个位是零：4*1
个位不是零：3*2

4.6.1 字符串倒序输出

给定一个字符串，然后将字符串倒序输出。
输入：输入一个字符串 S(2<strlen(S)<100)
输出：将字符串 S 倒序输出

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main()
{
    string s;
    getline(cin,s);
    reverse(s.begin(),s.end());
    cout<<s;
     return 0;
}
```

4.6.2 剪绳子

一条绳子从中间剪一刀可以剪成两段绳子；如果对折 1 次，中间剪一刀可以剪出 3 段绳子；如果连续对折 2 次，中间剪一刀可以剪出 5 段绳子；那么，连续对折 n 次，中间剪一刀可以剪出多少段绳子？

通过编写程序，在给定绳子对折次数，计算出中间剪一刀后可剪出绳子的段数。

输入

输入一个正整数口(1 <n< 20)作为绳子对折的次数

输出

输出一个正整数，表示对折 n 次后的绳子中间剪一刀可以剪出绳子的段数

输入样例 3

输出样例 9

```
#include <iostream>
using namespace std;
int main()
{
    int n;
cin>>n;
    cout<<(n-1)*2+1<<endl;
     return 0;
}
```

4.6.3 求和

合数指自然数中除了能被 1 和它本身整除外，还能被其他数(0 除外)整除的数。最小的合数是 4。

如：合数 4 既可以被 1 和 4 整除，还能被 2 整除。

给定一个正整数 N,计算出 4 到 N 之间所有合数的和。

例如：N 等于 7,其中 4 到 N 之间合数有 4、6,所有合数和等于 10 (4+6=10)

输入：输入一个正整数 N(4<N<101)

输出： 输出一个整数，表示 4 到 N 之间(包含 4 和 N)所有合数的和

输入样例 7

输出样例 10

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
```

```
{
    int n;cin>>n;
    int ans=0;
    for(int i=4;i<=n;i++){
        for(int j=2;j<=i/2;j++)
            if(i%j==0) {ans+=i;break;}
    }
    cout<<ans<<endl;
    return 0;
}
```

4.6.4 求和比较（dp）

小蓝在学习 C++数组时，突发奇想想知道如果将一个连续的正整数数组拆分成两个子数组，然后对拆分出的两个子数组求和并做差，且差值正好等于一个固定的正整数，像这样同一连续的正整数数组拆分方案有多少种。

我们一起帮助小蓝设计一下规则：

第一给出两个正整数 N 和 M;

第二从 1 到 N 组成一个连续正整数数组 A（A={1，2，3，4.…..N}）;

第三将数组 A 拆分成两个子数组 A1、A2 （1.两个子数组中不能出现相同的数;2.子数组中的数字可以是连续的也可以是不连续的;3.拆分出的两组子数组的元素个数可以不同，但总数量等于 A 数组元素个数）;

第四对 A1、A2 两个子数组分别求和;

第五对 A1、A2 两个子数组的和做差（大的数字减去小的数字）;

第六如果差值正好等于固定值 M，则判定此拆分方案成立。

如：N=5，M=1，连续正整数数组 A={1，2，3，4，5}。

符合条件的拆分方案有 3 种：

A1={1，2，4}，A2={3，5}，其中 A1 的和为 7，A2 的和为 8，和的差值等于 1

A1={1，3，4}，A2={2，5}，其中 A1 的和为 8，A2 的和为 7，和的差值等于 1

A1={3，4}，A2={1，2，5}，其中 A1 的和为 7，A2 的和为 8，和的差值等于 1

输入

分别输入两个正整数 N（3<N<30）和 M（0≤M≤<500）两个正整数由一个空格隔开

输出

输出一个正整数，表示 1 到 N （包含 1 和 N）连续的正整数数组中有多少种方案，使得拆分的两个子数组部分和的差值等于 M

输出样例 1

输入样例 1 5 1 3

```
#include <iostream>
#include <cstdio>
using namespace std;
int f[35][35];
int main()
{
    int n,m;
    cin>>n>>m;
    int sum=0;
    for(int i=1;i<=n;i++)sum+=i;
    sum-=m;
    if(sum%2!=0||sum<0){
        cout<<0<<endl;return 0;
    }
    /*设 x 为 A1 的和 设 y 为 A2 的和
    则有：x+y=sum  y-x=m
    则：x=(sum-m)/2  y=x+m
    */
    int x=sum/2;
    int y=x+m;
    /*f[i][j]前 i 个数字和恰好等于 j 的方案数
    */
    f[1][1]=1;
    for(int i=2;i<=n;i++){
        for(int j=1;j<=y;j++){
            f[i][j]=f[i-1][j];
            if(i==j) f[i][j]++;
            if(i<j) f[i][j]+=f[i-1][j-i];
            cout<<f[i][j]<<" ";
        }
        cout<<endl;
    }
    //m==0 时 有 x、y 互换的重复
    if(m==0) cout<<f[n][y]/2<<endl;
    else cout<<f[n][y]<<endl;
    return 0;
}
```

4.6.5 最大价值（dp）

一名种菜的农民伯伯，需要在给定的时间内完成种菜，现有 m 种不同的蔬菜提供给农民伯伯选择，且每种蔬菜种植花费的时间不同，每种蔬菜成熟后售卖的价值也不同。要求：

1. 在限定的总时间内进行蔬菜种植，并且种植蔬菜的种类不能超出限制的数量;

2.选择最优的种植方案使得蔬菜成熟后售卖的总价值最大（可选择不同的蔬菜种植）。

例如：

给定的总时间限制为 55，种植蔬菜的种类限制为 3;

3 种蔬菜，种菜的花费时间及售卖价格分别为：第一种 21 和 9，第二种 20 和 2，第三种 30 和 21。

最优的种植方案是选择种植第一种和第二种，两种蔬菜种植总时间 30+21。未超过总时间阳制 55。所种植蔬菜为两种， 也未超讨种类限制的 3 种最大总价值为 9+21=30，这个方案是最优的。

输入

第一行输入两个正整数 t（1≤t ≤600）和 m（1≤m ≤ 50），用一个空格隔开，t 代表种菜总时间限制，m 代表最多可种植蔬菜种类的限制接下来的 m 行每行输入两个正整数 t1（1<t1<101）和 p（1<p<101）且用一个空格隔开，t1 表示每种蔬菜种植需要花费的时间，p 表示对应蔬菜成熟后售卖的价值。

输出

输出一个正整数，表示选择最优的种植方案后，蔬菜成熟后售卖的最大总价值

输入样例 1 30

55　3

21　9

20　2

30　21

输出样例 30

```
#include<cstdio>
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;
int t,m;
struct greens{
    int tt,p;
}a[605];
int f[55][605];
```

```
int main()
{
    cin>>t>>m;
    for(int i=1;i<=m;i++)
        cin>>a[i].tt>>a[i].p;
    for(int i=1;i<=m;i++)
        for(int j=1;j<=t;j++){
            f[i][j]=f[i-1][j];
            if(j>=a[i].tt)
                f[i][j]=max(f[i][j],f[i-1][j-a[i].tt]+a[i].p);
        }
    cout<<f[m][t]<<endl;
    return 0;
}
```

4.6.6 精灵王国

在一个矩阵精灵王国里有两个精灵，一个叫黑精灵，个叫白精灵。他们住在一个 N*M 的矩阵方格中的不同位置，黑精灵住在矩阵方格的左上角方格里（1，1），白精灵住在矩阵方格的右下角方格里（N，M）。

在这个矩阵方格单还有一对可穿越的门，这对穿越门的位置不固定，位置可变换 （穿越门不会出现在矩阵方格左上角和右下角位置，也不会重善出现，有且只有一对）。穿越门的功能是当进入其中一扇门的位置后可直接穿越到另一扇门的位置。

如下图所示：

一天黑精灵要去白精灵家做客，需要穿过方格矩阵到达白精灵家，穿行矩阵方格要求：

1.每次只能走一个方格，可以向上、向下、向左、向右行走;

2.每走一个方格计为 1 步，但从一扇穿越门穿越到另一扇穿越门不记步数（从方格走到穿越门和从穿越门到其他方格都计 1 步）;

3.可借助穿越门去白精灵家（可减少行走步数）。

为了尽快到达白精灵家，请你帮助黑精灵找一条最短路线，并且计算出最短路线的步数。

例如：

给出一个 3*4 矩阵方格，并给出第一个穿越门的标位置 N1. M1（2.3） ，第二个穿越门的坐标位置 N2，M2（3.1）已知黑精灵初始坐标位置左上角（1，1），白精灵坐标位置右下角（N，M）。

假设用两个大写字母"D"表示矩阵方格中穿越门位置，1 代表黑精灵，2 代表白精灵，用数字 0 表示剩余矩阵方格。

如下图所示：

按照穿行矩阵方格要求为左上角方格的黑精灵到右下角方格白精灵家找一条最短路线,计算出最短路线的步数。
路线：从黑精灵初始位置（1，1）到正下方方格（2，1）走 1 步，正下方方格（2，1）到其下方穿越门（3，1）"D"走 1 步，然后穿越到另一扇穿越门（2，3）向正下方（3，3）走 1 步，最后到达白精灵家（3，4）需要走 1 步，故最短路线需要 4 步。
输入
第一行输入两个以一个空格隔开的正整数 N（2<N<101），M（2<M<101），分别表示 N 行 M 列的方格矩阵;
接下来第二行输入两个以一个空格隔开的正整数：N1（N1≤N），M1（M1≤M），代表第一个穿越门位于第 N1 行第 M1 列;接下来第三行输入两个以一个空格隔开的正整数：N2（N2≤N），M2（M2≤M），代表第二个穿越门位于第 N2 行第 M2 列;
注意：两个穿越门位置不能重叠，即不能同时满足 N1=N2 和 M1=M2;两个穿越门位置也不能位于左上角（1，1）和右下角（M，N）;第一个穿越门位置要在第二个穿越门前边或者上边。
输出
输出一个整数,表示黑精灵去白精灵家最短路线需要走多少步(可借助穿越门,减少步数),如果没有能到达白精灵家的路线或者其他情况统一输出数字"0"。
输出样例 1
输入样例 1 4
3 4 2 3 3 1

```
#include<cstdio>
#include<iostream>
#include<cstring>
#include<algorithm>
using namespace std;
int main()
{
    int n,m,n1,m1,n2,m2;
    cin>>n>>m>>n1>>m1>>n2>>m2;
    int s1=(n-1)+(m-1);
    int s2=(n1-1)+(m1-1)+(n-n2)+(m-m2);
    int s3=(n2-1)+(m2-1)+(n-n1)+(m-m1);
    int s=min(s1,min(s2,s3));
    cout<<s<<endl;
    return 0;
}
```

4.7***第 12 届蓝桥杯青少组省赛 C++中高级

单选题 30.0 分

第一题：下列符号中哪个在 C++中表示行注释（D）。

A. ！　　B. #　　C.]　　D. //

第二题：每个 C++程序都必须有且仅有一个（C）。

A. 函数　　B. 预处理命令　　C. 主函数　　D. 语句

第三题：下列字特串中不可以用作 C++变量名称的是（B）。

A. str123　　B. int　　C. _6666　　D. name

第四题：二进制加法 10010100+110010 的和为（A）。

A. 11000110　　B. 10100110　　C. 10110110　　D. 11100110

第五题：对于 int *pa[5]；的描述中，正确的是（D）。

A. pa 是一个指向数组的指针，所指向的数组是 5 个 int 型元素

B. pa 是一个指向某数组中第 5 个元素的指针，该元素是 int 型变量

C. pa[5]表示数组的第 5 个元素的值，是 int 型的值

D. pa 是一个具有 5 个元素的指针数组，每个元素是一个 int 型指针

4.7.1 整除

输入一个正整数 N（N<100），输出 1 到 N（包含 1 和 N）之间所有可以被 7 整除的正整数，且输出的正整数之间以一个空格隔开。

输入描述

输入一个正整数 N（N<100）

输出描述

输出可以被 7 整除的正整数，且输出的正整数之间以一个空格隔开

样例输入

15

样例输出

7 14

提示

评分标准：

10 分：能正确输出一组数据：

10 分：能正确输出两组数据；

10 分：能正确输出三组数据。

4.7.2 求和

有一堆砖，需要按照一定规律进行堆放，具体堆放规律如下：
顶层放 1 块砖，第二层放 3 块砖，第三层放 6 块砖，第四层放 10 块砖，依此类推，每一层砖块的数量为上一层砖块数量加上本层的层数。
例如第五层为 10+5=15。
输入砖块堆放的总层数，按照以上规律，求出砖块的总数。
题目描述：
输入一个正整数 N（3<N<1000）作为砖块堆放的总层数，按照“提示信息”中的堆放规律，输出砖块的总数。
例如：输入为 3，总层数为 3 层的砖块堆放一共有 1+3+6=10 块砖，则输出 10。
输入描述
输入一个正整数 N（3<N<1000）
输出描述
输出砖块的总数
样例输入
3
样例输出
10
提示
评分标准：
10 分：能正确输出一组数据：
10 分：能正确输出两组数据；
20 分：能正确输出三组数据。

4.7.3 排序

有 N 个瓶子，编号为从 1 到 N，这 N 个瓶子乱序排成一排。现在想对瓶子进行从小到大排列，要求每次拿出 2 个瓶子进行互换，经过若干次互换后，使得瓶子按编号从小到大排序，但这样的互换方式有多种，每种互换方式的互换次数也不相同。请你通过编程计算出最少经过几次互换可以使瓶子按编号从小到大排序。
题目描述：
第一行输入瓶子的个数 N（2<N<100），第二行输入目前瓶子编号的排序情况，输出最少经过几次互换可以使瓶子按编号从小到大排序。
例如第一行输入的 N 为 5，第二行输入 N 个正整数为 21354，第一次拿出 2 和 1 进行互换，第二次拿出 5 和 4 进行互换，最少需要两次互换。

输入描述
第一行输入一个正整数 N 表示瓶子的个数（2<N<100）
第二行输入 N 个正整数，之间用一个空格分开，表示瓶子编号目前的排序情况
输出描述
输出最少经过几次互换可以使瓶子按编号从小到大排序
样例输入
5
2 1 3 5 4
样例输出
2
提示
评分标准：
10 分：能正确输出一组数据：
10 分：能正确输出两组数据；
15 分：能正确输出三组数据；
15 分：能正确输出四组数据。

4.7.4 推算

小蓝是一名计算机极客，他在记录一些重要的日子时从不注明年月日，而是用一个整数替代，比如 4532，后来人们知道，那个整数就是日期，这个整数表示的日期就是他出生后的第几天。
他出生于：1999-04-30
例如他的日记里记录着获得蓝桥杯国寒总冠军的日子为 7856 这个整数，可以推断出这一天是 2020-10-31，现在需要请你计算出小蓝日记中其他整数对应的日期。
注意：输出的日期格式：yyyy-mm-dd，如：2020-03-21（月和日小于 10 的需要在月和日前补 0）
题目描述：
输入一个整数 n（5<n<30000）作为日记中记录的整数，输出这个整数对应的日期（注：按日期格式输出）。
例如：1999-04-30 为第 1 天，1999-05-01 为第 2 天。
输入描述
输入一个正整数 n（5<n<30000）
输出描述
输出这个整数对应的日期
样例输入

10

样例输出

1999-05-09

```
#include <iostream>
using namespace std;
bool run_n(int y){
    if(y%400==0||(y%4==0&&y%100!=0)) return true;
    return false;
}
int mo_d(int y,int m){
    if(m==1||m==3||m==5||m==7||m==8||m==10||m==12)
        return 31;
    else if(m==4||m==6||m==9||m==11) return 30;
    else if(m==2)
        if(run_n(y)) return 29;
        else return 28;
}
int main()
{
    int ye=1999,moth=5,day=1;
    int n;
    cin>>n;
    n-=2;
    while(1){
        int t=mo_d(ye,moth);
        if(n>t) {
            n-=t;moth++;
            if(moth>12) {
                moth=1;ye++;
            }
        }
        else {
            day+=n;break;
        }
    }
    cout<<ye<<"-"<<moth<<"-"<<day;
    return 0;
```

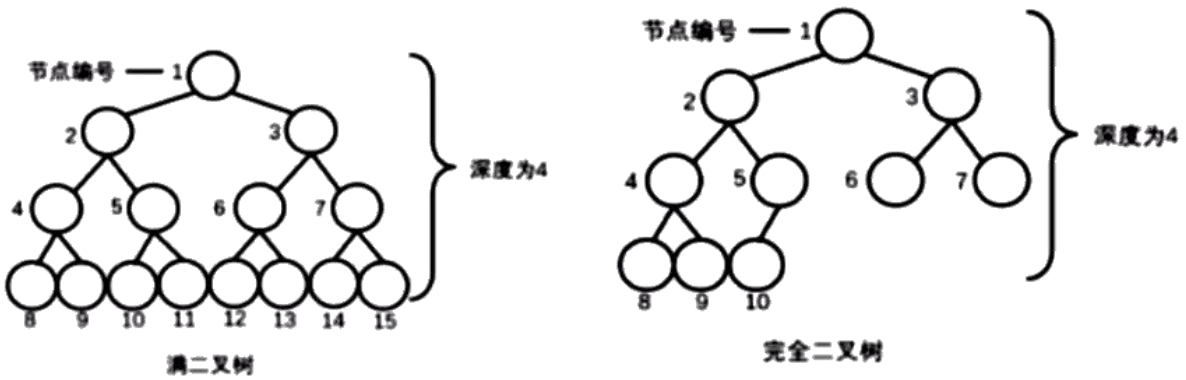
}

4.7.5 可逆素数

素数：素数就是质数，是一个大于 1 的自然数，且除了 1 和它本身外，不能被其他自然数整除的数。也就是说，除了 1 和该数本身以外不再有其他的因数的数被称为素数。最小的素数是 2，1 不是素数。可逆素数：是将一个素数的各个位置的数字顺序倒过来构成的反序数仍是素数。
例如：2，13，167 顺序或者反序都是素数
题目描述：
输入一个正整故 N（2<=N<10001），输出 2 到 N（包含 2 和 N）之间共有多少个可逆素数。
例如 2 到 15 之间共有 6 个可逆素数，分别为 2，3，5，7，11.13
输入描述
输入一个正整数 N（2≤N<10001）
输出描述
输出 2 到 N（包含 2 和 N）之间共有多少个可逆素数
样例输入
15
样例输出
6

4.7.6 满二叉树

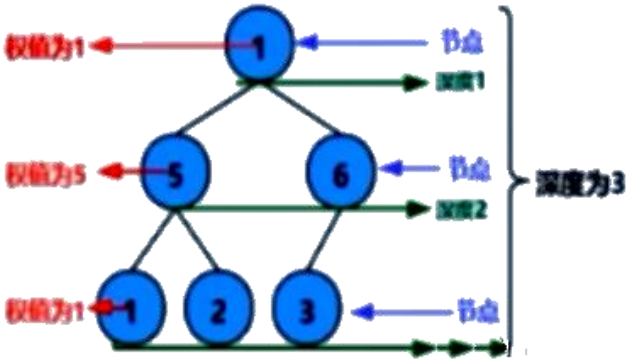
满二叉树：一棵二叉树，如果每一层的节点数都达到最大值，则这个二叉树就是满二叉树。即如果一棵二叉树的层数为 K，且结点总数是（2^k）-1，那么它就是满二叉树。
完全二叉树：一棵深度为 K 的二叉树，除第 K 层外，其他各层（1 至 k-1 层）的节点数都达到最大值，目第 K 层的所有节点都连续集中在左边，那么它就是完全二叉树。
节点：包含一个数据元素及若干指向子树分支的信息。
权值：对节点赋予的有意义的数量值。
深度：也被称为树的高度，树中所有节点的层次最大值称为树的深度，例如下图中的二叉树深度都为 4。



编程实现：

给出一棵包含 n 个节点的完全二叉树，节点按照从上到下、从左到右的顺序依次排序，每个节点上都有一个权值，如下图现在需要将同一深度节点的权值加在一起，然后比较每个深度的权值之和，输出权值之和最大的深度值。如果有多个深度的权值之和相同，则输出其中最小的深度（如：深度 2 权值之和为 5，深度 3 权值之和也为 5，则输出 2）。

注：根的深度为 1



第一行输入完全二叉树节点的总数量 n（5<n<101），第二行输入 n 个正整数作为每个节点的权值。输出权值之和最大的深度值（如果有多个深度的权值之和相同则输出其中最小的深度值）。

例如：上图的二叉树，第一行输入为 6，第二行输入为 1 5 6 1 2 3。深度 1 的权值之和为 1，深度 2 的权值之和为 11，深度 3 的权值之和为 6。其中深度 2 的权值之和最大，则输出 2。

输入描述

第一行输入一个正整数 n（5<n<101）作为节点的总数量

第二行输入 n 个正整数，且 n 个正整数之间以一个空格隔开

输出描述

输出权值之和最大的深度值

样例输入 1

6

1 5 6 1 2 3

样例输出 1

2

样例输入 2
8
1 5 6 1 2 3 4 100
样例输出 2
4

4.8***第十二届蓝桥省赛 C++大学生组 青少组省赛参考

4.8.1 空间

小蓝准备用 256MB 的内存空间开一个数组，数组的每个元素都是 32 位二进制整数，如果不考虑程序占用的空间和维护内存需要的辅助空间，请问 256MB 的空间可以存储多少个 32 位二进制整数？
256MB 换算成 B 就是 256*1024*1024
转换成位有 256*1024*1024*8
32 位整数就是 256*1024*1024*8/32

4.8.2 卡片

小蓝有很多数字卡片，每张卡片上都是数字 0 到 9。
小蓝准备用这些卡片来拼一些数，他想从 1 开始拼出正整数，每拼一个，就保存起来，卡片就不能用来拼其它数了。
小蓝想知道自己能从 1 拼到多少。
例如，当小蓝有 30 张卡片，其中 0 到 9 各 3 张，则小蓝可以拼出 1 到 10，但是拼 11 时卡片 1 已经只有一张了，不够拼出 11。
现在小蓝手里有 0 到 9 的卡片各 2021 张，共 20210 张，请问小蓝可以从 1 拼到多少？
提示：建议使用计算机编程解决问题。
开个数组存下 0-9 卡牌的数量，
然后从 1 开始往后遍历
每次都分解当前遍历的数
如果出现卡牌数量为零
说明当前数无法拼出
直接输出上一个数

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int a[11];
int main(){
    for(int i=0;i<10;i++)
        a[i]=2021;
    int cnt=1;
    while(1){
        int x=cnt;
        while(x){
            if(a[x%10]==0){
                cout<<cnt-1<<endl;
                return 0;
            }
            a[x%10]--;
            x/=10;
        }
        cnt++;
    }
}
//3181
```

4.8.3 货物摆放

小蓝有一个超大的仓库，可以摆放很多货物。现在，小蓝有 n 箱货物要摆放在仓库，每箱货物都是规则的正方体。小蓝规定了长、宽、高三个互相垂直的方向，每箱货物的边都必须严格平行于长、宽、高。

小蓝希望所有的货物最终摆成一个大的立方体。即在长、宽、高的方向上分别堆 L、W、H 的货物，满足 n = L × W × H。

给定 n，请问有多少种堆放货物的方案满足要求。

例如，当 n = 4 时，有以下 6 种方案：1×1×4、1×2×2、1×4×1、2×1×2、 2 × 2 × 1、4 × 1 × 1。

请问，当 n = 2021041820210418 （注意有 16 位数字）时，总共有多少种方案？

先质因数分解得到

2 1

3 3

17 1

131 1

2857 1

5882353 1

对于 2、17、131、2857、5882353，有 3^5=243 种方案

对于 3， 3、3、3 时有 1 种，1、3、9 有 6 种，1、1、27 有 3 种，合计 10 种

所以总方案数就是 243*10=2430

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
vector<int> prime;
int num[10000005];
bool vis[10000005];
void shai(){
    memset(vis,0,sizeof(vis));
    for(int i=2;i<=10000000;i++){
        if(!vis[i]){
            prime.push_back(i);
            for(int j=i+i;j<=10000000;j+=i){
                vis[j]=1;
            }
        }
    }
    return;
}
vector<int> v;
int main(){
    shai();
    ll n=2021041820210418;
    for(int i=0;i<prime.size();i++){
        num[i]=0;
        while(n%prime[i]==0){
            n/=prime[i];
            num[i]++;
        }
```

```
        if(num[i]!=0)
            v.push_back(i);
    }
    //ll check=1;
    for(int i=0;i<v.size();i++){
        cout<<prime[v[i]]<<' '<<num[v[i]]<<endl;
        //for(int j=1;j<=num[v[i]];j++)
        //check*=prime[v[i]];
    }
    //cout<<check<<endl;

}
//2430
```

4.8.4 时间显示

小蓝要和朋友合作开发一个时间显示的网站。在服务器上，朋友已经获取了当前的时间，用一个整数表示，值为从 1970 年 1 月 1 日 00:00:00 到当前时刻经过的毫秒数。
现在，小蓝要在客户端显示出这个时间。小蓝不用显示出年月日，只需要显示出时分秒即可，毫秒也不用显示，直接舍去即可。
给定一个用整数表示的时间，请将这个时间对应的时分秒输出。
【输入格式】
输入一行包含一个整数，表示时间。
【输出格式】
输出时分秒表示的当前时间，格式形如 HH:MM:SS，其中 HH 表示时，值为 0 到 23，MM 表示分，值为 0 到 59，SS 表示秒，值为 0 到 59。时、分、秒不足两位时补前导 0。
【样例输入 1】
46800999
【样例输出 1】
13:00:00
【样例输入 2】
1618708103123
【样例输出 2】
01:08:23
【评测用例规模与约定】
对于所有评测用例，给定的时间为不超过 1018 的正整数。

一天的时间是 86400000ms
将输入的数据对 86400000 取模，
然后再除以 1000 就是从当天 00:00:00 到现在经过的秒数。
每 3600 秒是一个小时，然后剩余的时间每 60 秒是一分钟，最后剩余的就是秒。

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int main(){
    ll a;
    cin>>a;
    a%=86400000;
    a/=1000;
    int h=a/3600,m=(a%3600)/60,s=a%60;
    if(h>=10)
        cout<<h<<':';
    else
        cout<<'0'<<h<<':';
    if(m>=10)
        cout<<m<<':';
    else
        cout<<'0'<<m<<':';
    if(s>=10)
        cout<<s;
    else
        cout<<'0'<<s;
}
```

4.8.5 砝码称重

你有一架天平和 N 个砝码，这 N 个砝码重量依次是 W1,W2,・・・,WN。
请你计算一共可以称出多少种不同的重量？
注意砝码可以放在天平两边。
【输入格式】
输入的第一行包含一个整数 N。
第二行包含 N 个整数：W1,W2,W3,・・・,WN。
【输出格式】
输出一个整数代表答案。

【样例输入】

3

1 4 6

【样例输出】

10

【样例说明】

能称出的 10 种重量是：1、2、3、4、5、6、7、9、10、11。

1 = 1；

2 = 6 － 4 (天平一边放 6，另一边放 4)；

3 = 4 － 1；

4 = 4；

5 = 6 － 1；

6 = 6；

7 = 1 + 6；

9 = 4 + 6 － 1；

10 = 4 + 6；

11 = 1 + 4 + 6。

【评测用例规模与约定】

对于 50% 的评测用例，1 ≤ N ≤ 15。

对于所有评测用例，1 ≤ N ≤ 100，N 个砝码总重不超过 100000。

跑两遍 01 背包，

第一遍是加砝码，第二遍是减砝码。

（对于减砝码

如果已经加上了当前的砝码，则相当于把砝码拿下来

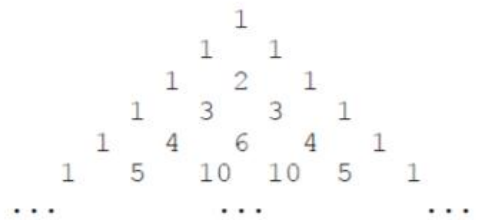
如果没有加当前的砝码，则相当于把砝码放在天平另一边

因此不会冲突）

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int dp[100005];
int w[105];
int main(){
    int n;
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>w[i];
    }
```

```
    memset(dp,0,sizeof(dp));
    dp[0]=1;
    for(int i=1;i<=n;i++){
        for(int j=100000;j>=w[i];j--){
            dp[j]=max(dp[j],dp[j-w[i]]);
        }
    }
    for(int i=1;i<=n;i++){
        int siz=100000-w[i];
        for(int j=1;j<=siz;j++){
            dp[j]=max(dp[j],dp[j+w[i]]);
        }
    }
    int ans=0;
    for(int i=1;i<=100000;i++){
        ans+=dp[i];
    }
    cout<<ans<<endl;
}
```

4.8.6 杨辉三角形



下面的图形是著名的杨辉三角形：

如果我们按从上到下、从左到右的顺序把所有数排成一列，可以得到如下数列：

1, 1, 1, 1, 2, 1, 1, 3, 3, 1, 1, 4, 6, 4, 1, …

给定一个正整数 N，请你输出数列中第一次出现 N 是在第几个数？

【输入格式】

输入一个整数 N。

【输出格式】

输出一个整数代表答案。

【样例输入】

6

【样例输出】

13

【评测用例规模与约定】

对于 20% 的评测用例，1 ≤ N ≤ 10;

对于所有评测用例，1 ≤ N ≤ 1000000000。

二项式定理，对于C(3，n)，当n等于2000时，C(3，2000)＞1e9

因此只需要算到第2000行就好了，剩下的再算C(1，n)和C(2，n)就好了。

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int a[2005][2005];
int main(){
    ll N;
    cin>>N;
    memset(a,0,sizeof(a));

    a[0][0]=1;
    for(int i=1;i<2005;i++){
        for(int j=1;j<=i;j++){
            a[i][j]=a[i-1][j]+a[i-1][j-1];
            if(a[i][j]==N){
                cout<<i*(i-1)/2+j<<endl;
                return 0;
            }
        }
    }
    //如果上面的没找到，说明只有C(1,n)和C(2,n)满足了
    //n*(n-1)/2==N
    ll n=sqrt(N*2)+1;
    if(n*(n-1)/2==N){
         //C(2,n)
         cout<<n*(n+1)/2+3<<endl;
    }else{
        //C(1,n)
```

```
        cout<<N*(N+1)/2+2<<endl;
    }
}
```

4.8.7 双向排序

给定序列 (a1, a2, ・ ・ ・ , an) = (1, 2, ・ ・ ・ , n)，即 ai = i。
小蓝将对这个序列进行 m 次操作，每次可能是将 a1, a2, ・ ・ ・ , aqi 降序排列，或者将 aqi , aqi+1, ・ ・ ・ , an 升序排列。
请求出操作完成后的序列。
【输入格式】
输入的第一行包含两个整数 n, m，分别表示序列的长度和操作次数。
接下来 m 行描述对序列的操作，其中第 i 行包含两个整数 pi, qi 表示操作类型和参数。当 pi = 0 时，表示将 a1, a2, ・ ・ ・ , aqi 降序排列；当 pi = 1 时，表示
将 aqi , aqi+1, ・ ・ ・ , an 升序排列。
【输出格式】
输出一行，包含 n 个整数，相邻的整数之间使用一个空格分隔，表示操作完成后的序列。
【样例输入】
3 3
0 3
1 2
0 2
【样例输出】
3 1 2
【样例说明】
原数列为 (1, 2, 3)。 第 1 步后为 (3, 2, 1)。 第 2 步后为 (3, 1, 2)。 第 3 步后为 (3, 1, 2)。与第 2 步操作后相同，因为前两个数已经是降序了。
【评测用例规模与约定】
对于 30% 的评测用例，n, m ≤ 1000;
对于 60% 的评测用例，n, m ≤ 5000;
对于所有评测用例，1 ≤ n, m ≤ 100000，0 ≤ ai ≤ 1，1 ≤ bi ≤ n。

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
/*
```

```
int main(){

}
*/

//暴力分 n*n*logn
bool cmp(int x,int y){
    return x>y;
}
int a[100005];
int main(){
    int n,m;
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        a[i]=i;
    while(m--){//1e5
        int p,q;
        cin>>p>>q;
        //nlogn
        if(p==0){
            sort(a+1,a+q+1,cmp);
        }else{
            sort(a+q,a+n+1);
        }
        for(int i=1;i<=n;i++)
        cout<<a[i]<<' ';
        cout<<endl;
    }
    for(int i=1;i<=n;i++)
        cout<<a[i]<<' ';
}
```

4.8.8 趣数（动态规划题目 省赛难度）

如 126543 这样的数，6 是其中最大的，在 6 的左边越来越大，在 6 的右边越来越小，并且在这个数中，1、2、3、4、5、6 各出现一次，我们把这样的数叫 6 的趣数。

编程实现

输入数字 N（3<=N<=9)

输出 N 的趣数，数字间用空格分开。

输入样例

4

输出样例

1243 1342 1432 2341 2431 3421

```
#include <iostream>
using namespace std;
int a[10],book[10],n;
bool judge()
{
    int i=1;
    if(a[i]==n||a[n]==n) return false;
    while(a[i]<n&&a[i]<a[i+1])i++;
    i++;
    while(a[i]<n&&a[i]<a[i-1]&&i<=n)i++;
    i--;
    if(i==n) return true;
    else return false;
}
void dfs(int step)
{
    int i;
    while(step==n+1){
        if(judge()){
            for(int i=1;i<=n;i++)
                cout<<a[i];
            cout<<" ";
        }
        return ;
    }
    for(i=1;i<=n;i++){
```

```
        if(book[i]==0){
            a[step]=i;
            book[i]=1;
            dfs(step+1);
            book[i]=0;
        }
    }
}
int main(int argc, char *argv[])
{
    cin>>n;
    dfs(1);
    return 0;
}
```

4.8.9 括号合法组合

循环以及循环的嵌套，是同学们编写程序时常见的操作。如果用一对括号来代表一个循环的话，那么三个循环出现的合法组合有 5 种，分别为：“{}{}{} ”、“{{{}}}”、“{{}{}}”、“{{}}{}”、“{}{{}}”。

输入：一个数字 n（1<=n<=10），代表循环的个数。

输出：输出 n 个循环的合法组及组合个数。

样例输入 1：

2

样例输出 1：

{}{}

{{}}

2

样例输入 2：

3

样例输出 2：

{}{}{}

{{}}{}

{}{{}}

{{}{}}

{{{}}}

5

```
#include <cstdio>
#include <cstring>
#include <iostream>
using namespace std;
int n,lf,rt,s,tj,tmp;
int aa[65];
int main( )
{
    cin>>n;
    int t=1,a=0,b=0;
    tj=0;
    for(int i=1;i<=2*n;i++){
        if(i%2==0)a+=t;
        if(i>n)b+=t;
        t*=2;
    }
//  cout<<a<<" "<<b<<endl;
    for(int j=a;j<=b;j++){
        t=j;
        lf=rt=0;s=1;
        memset(aa,0,sizeof(aa));
        while(t>0){
            if(t&1){lf++;aa[s]=1;}
            else {rt++;aa[s]=0;}
            if(lf>rt)break;
            s++;
            t=t>>1;
        }
        if(aa[1]==0&&lf==rt&&lf==n){
            tj++;
            for(int i=1;i<s;i++){
                if(aa[i]==0) cout<<"{";
                else cout<<"}";
            }
            cout<<"\n";  //cout<<endl;
        }
    }
```

```
    cout<<tj<<endl; //printf("%d\n",tj);
    return 0;
}
```

4.9***第十一届蓝桥杯青少年组 C++国赛高级组

单选题 30.0 分

一．在数组中，数组名表示（A）.

A.数组第 1 个元素的首地址　　B.数组第 2 个元素的首地址

C.数组所有元素的首地址　　D.数组最后 1 个元素的首地址

二．下列叙述中正确的是（A）.

A.顺序存储结构的存储一定是连续的，链式存储结构的存储空间不一定是连续的

B.顺序存储结构只针对线性结构，链式存储结构只针对非线性结构

C.顺序存储结构能存储有序表，链式存储结构不能存储有序表

D.链式存储结构比顺序存储结构节省存储空间

三．下列排序方法中，最坏情况下比较次数最少的是（D）.

A.冒泡排序　　B.简单的选择排序

C.直接插入排序　　D.堆排序

四．下列表达式正确的是（）.

A. 9++　　B. (x+y)++

C. c+++c+++c++　　D.++(a-b--)

五．针对以下代码，判断下列说法哪个是正确的（C）.

```
const char str1[]="abc";
const char str2[] ='abc";
const char *p1 ='abc";
const char *p2 ='abc";
```

A. str1 和 str2 地址不同，P1 和 P2 地址相同

B. str1 和 str2 地址相同，P1 和 P2 地址相同

C. str1 和 str2 地址不同，P1 和 P2 地址不同

D. str1 和 str2 地址相同，P1 和 P2 地址不同

4.9.1 求阶乘

阶乘定义：一个正整数的阶乘是所有小于及等于该数的正整数的乘积。

例如：3 的阶乘 6（也就是 1*2*3 的结果）

例如：5 的阶乘 120（也就是 12*3*4*5 的结果）

题目描述：输入一个正整数 N（3<=N<=20），输出 1 到 N 之间（包含 1 和 N）所有正整

数阶乘的和。
例如：输入为 3，1 的阶乘为 1，2 的阶乘为 2，3 的阶乘为 6，
1+2+6=9，则输出 9.
输入描述
输入一个正整数 N（3<=N<=20）
输出描述
输出 1 到 N 之间（包含 1 和 N）所有正整数阶乘的和
样例输入
3
样例输出
9

4.9.2 判断偶数

输入两个不相等的四位正整数 N（1000<=N<=9999）和 M（1000<=M<=9999），其中 N<M，中间以一个空格隔开，输出 N 与 M 之间（包含 N 和 M）所有满足要求的正整数且正整数之间以一个英文逗号隔开。
要求每个正整数的各个位上的数都为偶数（注：0 为偶数）
输入描述
在一行输入两个不相等的四位正整数 N 和 M（N 和 M 之间以一个空格隔开）
输出描述
输出 N 与 M 之间（包含 N 和 M）所有满足要求的正整数且正数之间以一个英文逗号隔开
样例输入
4000 4008
样例输出
4000，4002，4004，4006，4008

4.9.3 计数

输入一个正整数 n（1 <=n<= 1000），统计从 1 到 n 之间（包含 1 和 n）所有正整数中，0，1，2，3，4，5，6，7，8，9 的数字分别出现的次数，且按样例分行输出（按 0 到 9 顺序输出，英文逗号前为 0 到 9 的数字，逗号后为该数字出现的次数）。
例如：n 为 12，那么 1 到 n 之间所有的正整数有 1，2，3，4，5，6，7，8，9，10，11，12.
在 12 个正整数中数字 0 出现了 1 次数字 1 出现了 5 次数字 2 出现了 2 次数字 2，3，4，5，6，7，8，9 分别出现了 1 次。
输入描述

输入一个正整数 n
输出描述
0，0 出现的次数
1，1 出现的次数
2，2 出现的次数
……
9，9 出现的次数
样例输入
10
样例输出
0,1
1,2
2,1
3,1
4,1
5,1
6,1
7,1
8,1
9,1

4.9.4 找公共子串

分行输入两个字符串（2<=字符串长度<=100），找出两个字符串中最大的公共子串，然后将公共子串及公共子串的长度分行输出
例如：输入两个字串为 abcdab 和 baabcd，其最大的公共子串为
"abcd"，子串长度为 4.
输入描述
第一行输入一个字符串
第二行输入一个字符串
输出描述
第一行输出最大公共子串
第二行输出最大公共子串长度
样例输入
abcdab
baabcd
样例输出

abcd
4

4.9.5 最少问题

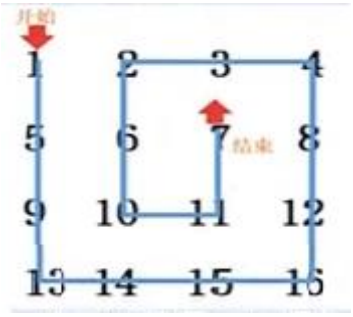
输入两个整数 n（0<n<100001）和 k（0<k<100001），通过对 n 连续进行加 1 或减 1 或乘以 2 这 3 种操作，使得 n 最后结果正好等于 k（同一种操作可以使用多次也可以不使用），要求最后输出最少的操作次数。
例如：n 为 5，k 为 17，通过减 1、乘以 2、乘以 2、加 1 四次操作得到 17，也就是 5-1=4，4*2=8、8*2=16，16+1=17.
输入描述
输入两个整数 n 和 k（n 和 k 之间以一个空格隔开）
输出描述
输出最少的操作次数
样例输入
5 17
样例输出
4

```
//最少问题
#include <iostream>
#include <cstdio>
#include <queue>
using namespace std;
struct node {
    int pos,step;
};
int v[100005];
queue<node> q;
int main()
{
    int n,k;
    cin>>n>>k;
    v[n]=1;
    q.push((node){n,0});
    while(!q.empty()){
        node t=q.front();
        q.pop();
```

```
        //cout<<t.pos<<" "<<t.step<<endl;
        if(t.pos==k){
            cout<<t.step<<endl;
            break;
        }
        if(t.pos+1<100005&&v[t.pos+1]==0){
            v[t.pos+1]=1;
            q.push((node){t.pos+1,t.step+1});
        }
        if(t.pos-1>=0&&v[t.pos-1]==0){
            v[t.pos-1]=1;
            q.push((node){t.pos-1,t.step+1});
        }
        if(t.pos*2<100005&&v[t.pos*2]==0){
            v[t.pos*2]=1;
            q.push((node){t.pos*2,t.step+1});
        }
    }
    return 0;
}
```

4.9.6 回形取数（源自蓝桥 VIP 题库）

回形取数是沿着一个数字矩阵的左上角向下开始移动取数当前方没有数字或者数字已经被取过就会左转继续移动取数当没有数可取时回形取数结束，如下图所示



回形取数结束后会产生一条线路图，也就是数字线路。

上路为：1，5，9，13，14，15，16，12，8，4，3，2，6，10，11，7

题目描述：

用户分行输入两个正整数（2<=正整数<=20），第一个数代表数字矩阵的行数，第二个数代表数字矩阵的列数，数字矩阵的数字为从 1 开始的正整数，根据回形取数规则将最终的

数字线路输出（数字线路中的每个数字之间需要有一个英文逗号隔开）。
例如：
输入的是 2 和 3 数字矩阵为：
1 2 3
4 5 6
数字线路为：1，4，5，6，3，2
输入的是 4 和 3 数字矩阵为：
1 2 3
4 5 6
7 8 9
10 11 12
数字线路为：1，4，7，10，11，12，9，6，3，2，5，8
注：数字矩阵不需要输入此处只为展示
输入描述
第一行输入一个正整数作为行数
第二行输入一个正整数作为列数
编出描述
根据回形取数规则将数字线路输出（数字线路中的每数字之间需要有一个英文逗号隔开）
样例输入
3
2
样例输出
1，3，5，6，4，2

```
//回形取数
#include <iostream>
#include <cstdio>
using namespace std;
int v[105][105],a[105][105];
int main()
{
    int n,m,k=0;
    int i,j;
    cin>>n>>m;
    for(i=0;i<n;i++)
        for(j=0;j<m;j++) a[i][j]=++k;
    i=k=1;j=0;
    cout<<a[0][0];
```

```
    v[0][0]=1;
    while(k<n*m){
        while (i<n && v[i][j]==0){
            cout<<",<<a[i][j];
            v[i][j]=1;i++;k++;
        }
        i--;j++;
        while(j<m && v[i][j]==0){
            cout<<",<<a[i][j];
            v[i][j]=1;
            j++;k++;
        }
        i--;j--;
        while (i>=0&&v[i][j]==0){
            cout<<",<<a[i][j];
            v[i][j]=1;
            i--;k++;
        }
        i++;j--;
        while(j>=0 && v[i][j]==0){
            cout<<",<<a[i][j];
            v[i][j]=1;
            j--;k++;
        }
        i++;j++;
    }
    return 0;
}
```

4.9.7 带分数

提示信息：

带分数是分数的一种形式，带分数包含两个部分：整数部分和真分数部分。

例如：$3\frac{3}{4}$ 前边的 3 为整数部分，后边 4 分之 3 为真分数部分。读作：三又四分之三.

例如：

5 用带分数形式可表示为：3+2956/1478，3+2956/1478，3+9562/4781，3+9712/4856

要求：带分数中，数字 1-9 分别出现且只出现一次。

5 的带分数形式有 4 种表示法.

输出这个正整数满足要求的带分数表示方法，一共有多少种

样例输入

100

样例输出

11

//带分数(暴力枚举)

```
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int n,cnt=0;
int p[]={1,2,3,4,5,6,7,8,9};
bool check(int x,int y){
    int a,b,c;
    a=b=c=0;
    for(int i=0;i<x;i++)a=a*10+p[i];
    for(int i=x;i<y;i++)b=b*10+p[i];
    for(int i=y;i<9;i++)c=c*10+p[i];
    if(b%c==0&&a+b/c==n) return true;
    else return false ;
}
int main()
{
    cin>>n;
    do{
        for(int i=1;i<7;i++)
            for(int j=i+1;j<8;j++)
                if(check(i,j)) cnt++;
    }while(next_permutation(p,p+9));
    cout<<cnt<<endl;
     return 0;
}
```

4.10 ***第十二届蓝桥杯青少年组国赛 C++中级组

4.10.1 加密

给定一个只含英文字母（英文字母含大小写字母）的字符串作为原始密码，按照规则将其加密，并输出加密后的密码。

加密原则：原始密码每一位上的字母，使用其在字母表中其后的第三个字母替代原来的字母。如：原始密码是 abC，a 的字母表中气候的第三个字母为 d，b 后的第三个字母为 e，C 后的第三个字母为 F，故加密后的密码为 deE。

注：

1、当原始密码当前为的字母为小写，加密后的密码为字母表中气候的第三个小写字母。当原始密码当前为的字母为大写，加密后的密码为字母表中气候的第三个大写字母。

2、原始密码中的英文字母不能为 x（X），y（Y），z（Z）。

输入描述

输入 N 众英文字母（包含大小写字母，但不包含大小写的义，V，Z，2<N<101）作为原始密码。

输出描述

按照加密规则输出一个字符串，作为加密后的密码。

输入样例

abC

输出样例

deF

```
#include <iostream>
using namespace std;
int main() {
    string s;
    int len;
    cin >> s;
    len = s.length();
    for (int i = 0; i < len; i++)
        cout << char(s[i] + 3); //字符串可以当数组访问
    return 0; //字符型加整型 cout 默认输出整型 用 char()强制转换回来
}
```

4.10.2 分解质因数

给定一个合数 N，将 N 分解质因数后，输出其质因数的个数。
比如输入：18，输出：3。 18=2*3*3

```
#include <iostream>
#include <cmath> //sqrt()开平方函数使用
using namespace std;
int main() {
    int n;
    cin >> n;
    int i, count = 0;
    for (i = 2; i <= sqrt(n); i++) {
        while (n % i == 0) {//能被整除就继续循环
            count++;
            n =n / i;
        }
    }
    cout << count;
    return 0;
}
```

4.10.3 开关门（纯模拟）

学校 N 间教室，每个教室 2 扇门，每扇门都有编号，分别为 1 到 2*N。
如下规则处理：1，打开所有门。第 2 次，编号为 2 的倍数相反处理。
如此类推，第 n 次处理后，有多少扇门为打开状态。
比如输入：2，输出：2。

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    bool a[2 * n + 1] = {0}; //0 关闭 1 打开
    int i, j, count = 0;
    for (i = 1; i <= n; i++) { //操作多少次
        for (j = 1; j <= 2 * n; j++) { //处理多少扇门
```

```
            if (j % i == 0) a[j] = !a[j]; //倍数关系相反处理
        }
    }
    for (i = 1; i <= 2 * n; i++) {
        if (a[i] == 1) count++;
    }
    cout << count;
    return 0;
}
```

4.10.4 买瓜子（完全背包 常考必会题型）

校庆，采购瓜子。资金 N（1<=n<=1000）元，M（1<= m<=30）种瓜子。
问最多能采购多少千克的瓜子？比如 N=80 元，M=2 种。第 1 种，每袋 18 元 10 千克；第 2 种，每袋 30 元 20 千克。
输入：
80 2
18 10
30 20
输出：
50
解题分析： 完全背包问题 模板题

```
#include <iostream>
using namespace std;
const int N = 1010;
int v[50], wt[1010], dp[50][1010];
using namespace std;
int main() {
    int w, n;
    cin >> w >> n;
    for (int i = 1; i <= n; i ++)
        cin >> wt[i] >> v[i];

    for (int i = 1; i <= n; i ++)
        for (int j = 1; j <= w ; j ++) {
            if (j >= wt[i])
                dp[i][j] = max(dp[i - 1][j], dp[i][j - wt[i]] + v[i]);
```

```
            else
                dp[i][j] = dp[i - 1][j]; //空间不够就和前面一样
            }
    cout << dp[n][w];
    return 0;
}
```

4.10.5 投篮（动态规划-最长下降子序列）

N 个篮筐，投球到篮筐。玩法如下：
规则 1，第 1 次投篮，可以投任意篮筐。
规则 2，第 2 次及以后，每次投上次后面的蓝筐。
规则 3，第 2 次及以后，每次只能投比上次更近的篮筐。
给出 N 个篮筐的距离和顺序，求最多能投进几个球。
如 3 个篮筐的距离和顺序为 130，200，55。有 3 种投法：
1）第 1 次投 130，那么投 130，55，最多投进 2 球。
2）第 2 次投 200，那么投 200，55，最多投进 2 球。
3）第 2 次投 55，那么投 55，最多投进 1 球。
因此，按照规则，最多可投进 2 球。
输入：
3
130 200 55
输出：
2

```
//投篮(动规-最长下降子序列)
#include <iostream>
#include <cstdio>
#include <algorithm>
using namespace std;
int n,a[1005],dp[1005];
int main()
{
    cin>>n;
    for(int i=0;i<n;i++) cin>>a[i];
    int maxn=0;
    for(int i=0;i<n;i++){
```

```
        dp[i]=1;
        for(int j=0;j<i;j++)
            if(a[i]<a[j]) dp[i]=max(dp[i],dp[j]+1);
        maxn=max(maxn,dp[i]);
    }
    cout<<maxn<<endl;
    return 0;
}
```

4.10.6 摘苹果

果园里有两种水果，苹果（1）和梨子（2），起点（6）也是苹果。
从起点开始，不能碰到梨子，求最多能摘到多少个苹果。
输入：
3 4
2 1 2 1
1 6 1 2
1 1 1 2
输出：
7

```
//摘苹果(标准深搜问题)
#include <iostream>
#include <cstdio>
using namespace std;
int n,m,a[105][105],tj=0;
int x1[4]={-1,1,0,0};
int y1[4]={0,0,-1,1};
int dfs(int x,int y){
    for(int i=0;i<4;i++){
        int xx=x+x1[i];
        int yy=y+y1[i];
        if(xx>=1 && xx<=n && yy>=1 && yy<=m && a[xx][yy]==1){
            a[xx][yy]=0;tj++;dfs(xx,yy);
        }
    }
}
int main()
```

```
{
    cin>>n>>m;
    int sx,sy;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>a[i][j];
            if(a[i][j]==6){
                a[i][j]=0;
                sx=i;sy=j;
            }
        }
    }
    tj++;
    dfs(sx,sy);
    cout<<tj<<endl;
    return 0;
}
```

5 蓝桥杯官网练习系统 VIP 题目题解

***（省赛高级组或国赛初级中级组必看）

5.1.1 2n 皇后问题

给定一个 n*n 的棋盘，棋盘中有一些位置不能放皇后。现在要向棋盘中放入 n 个黑皇后和 n 个白皇后，使任意的两个黑皇后都不在同一行、同一列或同一条对角线上，任意的两个白皇后都不在同一行、同一列或同一条对角线上。问总共有多少种放法？n 小于等于 8。

输入格式

输入的第一行为一个整数 n，表示棋盘的大小。

接下来 n 行，每行 n 个 0 或 1 的整数，如果一个整数为 1，表示对应的位置可以放皇后，如果一个整数为 0，表示对应的位置不可以放皇后。

输出格式

输出一个整数，表示总共有多少种放法。

样例输入

```
4
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
```

样例输出
2
样例输入
4
1 0 1 1
1 1 1 1
1 1 1 1
1 1 1 1
样例输出
0

```
#include <stdio.h>
int go(int* a, int* b, int* c, int m, int n) {
    if (m == n)
        return 1;

    int i, j, k, count = 0;
    for (i = 0; i < n; i++) {
        if (a[m * n + i] == 0)
            continue;
        for (j = 0; j < m; j++)
            if (b[j] == i || b[j] - i == j - m || b[j] + j == i + m)
                break;
        if (j == m) {
            b[m] = i;
            for (k = 0; k < n; k++) {
                if (a[m * n + k] == 0 || k == i)
                    continue;
                for (j = 0; j < m; j++)
                    if (c[j] == k || c[j] - k == j - m || c[j] + j == k + m)
                        break;
                if (j == m) {
                    c[m] = k;
                    count += go(a, b, c, m + 1, n);
                }
            }
        }
    }
    return count;
}

int main() {
    int a[64], b[8], c[8], i, j, n;
    scanf("%d", &n);
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
```

```
        scanf("%d", &a[i * n + j]);

    printf("%d", go(a, b, c, 0, n));
    return 0;
}
```

5.1.2 时间转换（取余数字字符混合输出）

给定一个以秒为单位的时间 t，要求用“<H>:<M>:<S>”的格式来表示这个时间。<H>表示时间，<M>表示分钟，而<S>表示秒，它们都是整数且没有前导的“0”。例如，若 t=0，则应输出是“0:0:0”；若 t=3661，则输出“1:1:1”。

输入格式

输入只有一行，是一个整数 t（0<=t<=86399）。

输出格式

输出只有一行，是以“<H>:<M>:<S>”的格式所表示的时间，不包括引号。

样例输入

0

样例输出

0:0:0

样例输入

5436

样例输出

1:30:36

```
#include <stdio.h>

int main()
{
    int n;
    scanf("%d", &n);
    printf("%d:%d:%d", n / 3600, n % 3600 / 60, n % 60);
    return 0;
}
```

5.1.3 分解质因数 (质数分解循环)

求出区间[a,b]中所有整数的质因数分解。

输入格式

输入两个整数 a，b。

输出格式

每行输出一个数的分解,形如 k=a1*a2*a3...(a1<=a2<=a3...,k 也是从小到大的)(具

体可看样例)
样例输入
3 10
样例输出
3=3
4=2*2
5=5
6=2*3
7=7
8=2*2*2
9=3*3
10=2*5
提示
先筛出所有素数，然后再分解。
数据规模和约定
2<=a<=b<=10000

```
#include <stdio.h>
#include <math.h>
int factor(n)
{
    int i, j = (int)sqrt(n);
    if (n % 2 == 0) return 2;
    for (i = 3; i <= j; i++)
        if (n % i == 0) return i;
    return n;
}
int main()
{
    int i, j, k, m, n;
    scanf("%d%d", &m, &n);
    for (i = m; i <= n; i++)
    {
        j = factor(i);
        k = i / j;
        printf("%d=%d", i, j);
        while (k > 1)
        {
```

```
            j = factor(k);
            k /= j;
            printf("*%d", j);
        }
        printf("\n");
    }
    return 0;
}
```

5.1.4 矩阵乘法 (二维数组循环矩阵)

给定一个 N 阶矩阵 A，输出 A 的 M 次幂（M 是非负整数）

例如：

A =

1 2

3 4

A 的 2 次幂

7 10

15 22

输入格式

第一行是一个正整数 N、M（1<=N<=30, 0<=M<=5），表示矩阵 A 的阶数和要求的幂数

接下来 N 行，每行 N 个绝对值不超过 10 的非负整数，描述矩阵 A 的值

输出格式

输出共 N 行，每行 N 个整数，表示 A 的 M 次幂所对应的矩阵。相邻的数之间用一个空格隔开

样例输入

2 2

1 2

3 4

样例输出

7 10

15 22

```
#include <stdio.h>
#include <iostream>
using namespace std;
```

```
int main()
{
    int a[30][30], b[30][30] = {0}, c[30][30];
    int i, j, k, l, m, n;
    scanf("%d%d", &n, &m);
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &a[i][j]);
    for (i = 0; i < n; i++)
        b[i][i] = 1;
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
        {
            for (k = 0; k < n; k++)
            {
                c[j][k] = 0;
                for (l = 0; l < n; l++)
                    c[j][k] += a[j][l] * b[l][k];
            }
        }
        for (j = 0; j < n; j++)
            for (k = 0; k < n; k++)
                b[j][k] = c[j][k];
    }
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
            printf("%d ", b[i][j]);
        printf("\n");
    }
    return 0;
}
```

5.1.5 完美的代价(贪心算法 回文)

回文串，是一种特殊的字符串，它从左往右读和从右往左读是一样的。小龙龙认为回

文串才是完美的。现在给你一个串，它不一定是回文的，请你计算最少的交换次数使得该串变成一个完美的回文串。

交换的定义是：交换两个相邻的字符

例如 mamad

第一次交换 ad : mamda

第二次交换 md : madma

第三次交换 ma : madam (回文！完美！)

输入格式

第一行是一个整数 N，表示接下来的字符串的长度(N <= 8000)

第二行是一个字符串，长度为 N.只包含小写字母

输出格式

如果可能，输出最少的交换次数。

否则输出 Impossible

样例输入

5

mamad

样例输出

3

```
#include <stdio.h>
#include <iostream>
using namespace std;
void swap(char* a, char* b) {
    char c = *a;
    *a = *b;
    *b = c;
}
int find(char* p, char* q) {
    char* o = q - 1;
    while (o > p && *o != *p) o--;
    return o - p;
}

int rfind(char*p, char* q) {
    char* o = q + 1;
    while (o < p && *o != *p) o++;
    return p - o;
}
```

```
int main() {
    int i, j, k, len, count[26] = {0};
    char str[8000];
    scanf("%d", &len);
    scanf("%s", str);
    for (i = 0; i < len; i++)
        count[str[i] - 'a']++;
    int odd = 0;
    for (i = 0; i < 26; i++)
        if (count[i] % 2 == 1)
            odd++;
    if (odd <= 1) {
        int change = 0;
        for (i = 0; i < len / 2; i++) {
            if (count[str[i] - 'a'] == 1)
                break;
            count[str[i] - 'a'] -= 2;
            k = find(str + i, str + len - i);
            for (j = i + k; j < len - i - 1; j++) {
                swap(str + j, str + j + 1);
                change++;
            }
        }
        if (i != len / 2) {
            for (i = len - i - 1; i > len / 2; i--) {
                k = rfind(str + i, str + len - i - 1);
                for (j = i - k; j > len - i - 1; j--) {
                    swap(str + j, str + j - 1);
                    change++;
                }
            }
        }
        printf("%d", change);
    } else {
        printf("Impossible");
    }
```

```
    return 0;
}
```

5.1.6 数的读法（模拟 判断函数）

Tom 教授正在给研究生讲授一门关于基因的课程，有一件事情让他颇为头疼：一条染色体上有成千上万个碱基对，它们从 0 开始编号，到几百万，几千万，甚至上亿。

比如说，在对学生讲解第 1234567009 号位置上的碱基时，光看着数字是很难准确的念出来的。

所以，他迫切地需要一个系统，然后当他输入 12 3456 7009 时，会给出相应的念法：

十二亿三千四百五十六万七千零九

用汉语拼音表示为

shi er yi san qian si bai wu shi liu wan qi qian ling jiu

这样他只需要照着念就可以了。

你的任务是帮他设计这样一个系统：给定一个阿拉伯数字串，你帮他按照中文读写的规范转为汉语拼音字串，相邻的两个音节用一个空格符格开。

注意必须严格按照规范，比如说“10010”读作“yi wan ling yi shi”而不是“yi wan ling shi”，“100000”读作“shi wan”而不是“yi shi wan”，“2000”读作“er qian”而不是“liang qian”。

输入格式

有一个数字串，数值大小不超过 2,000,000,000。

输出格式

是一个由小写英文字母，逗号和空格组成的字符串，表示该数的英文读法。

样例输入

1234567009

样例输出

shi er yi san qian si bai wu shi liu wan qi qian ling jiu

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main() {
    string num[] = {"ling", "yi", "er", "san", "si", "wu", "liu", "qi", "ba", "jiu"};
    string wei[] = {"shi", "bai", "qian", "wan", "yi"};
    string str[20];
    int n;
```

```
    scanf("%d", &n);
    int i = 0, j = 0, k, l[2] = {0};
    while (n > 0) {
        k = n % 10;
        n /= 10;
        if (k > 0) {
            if (i > 0) {
                if (i >= 4 && !l[i / 4 - 1]) l[i / 4 - 1] = 1, str[j++] = wei[i /
4 + 2];
                if (i % 4 != 0) str[j++] = wei[i % 4 - 1];
            }
            str[j++] = num[k];
        } else if (j > 0 && str[j - 1] != num[0]) {
            str[j++] = num[0];
        }
        i++;
    }
    if (!(str[j - 1] == num[1] && j > 1 && str[j - 2] == wei[0]))
        printf("%s ", str[j - 1]);
    for (i = j - 2; i >= 0; i--)
        printf("%s ", str[i]);
    return 0;
}
```

5.1.7 FJ 的字符串（字符串递归）

FJ 在沙盘上写了这样一些字符串：

A1 = “A”

A2 = “ABA”

A3 = “ABACABA”

A4 = “ABACABADABACABA”

… …

你能找出其中的规律并写所有的数列 AN 吗？

输入格式

仅有一个数：N ≤ 26。

输出格式

请输出相应的字符串 AN，以一个换行符结束。输出中不得含有多余的空格或换行、

回车符。
样例输入
3
样例输出
ABACABA
//warning: unused variable ‘j’ [-Wunused-variable]
发现未被使用的变量 j 声明变量未使用　程序提交前如果确认不使用该变量请删除，保证代码的整洁度。

```
#include <stdio.h>
#include <string.h>
#include <iostream>
using namespace std;
int main() {
    int i, n;
    char str[1024] = "A", str2[1024];
    scanf("%d", &n);
    for (i = 1; i < n; i++) {
        sprintf(str2, "%s%c%s", str, 'A' + i, str);
        strcpy(str, str2);
    }
    printf("%s", str);
    return 0;
}
```

//strcpy()函数：字符串复制（拷贝）函数
strcpy(a,b)将 b 数组中的字符串复制到 a 数组中

5.1.8 芯片测试(数组模拟)

有 n（2≤n≤20）块芯片，有好有坏，已知好芯片比坏芯片多。

每个芯片都能用来测试其他芯片。用好芯片测试其他芯片时，能正确给出被测试芯片是好还是坏。而用坏芯片测试其他芯片时，会随机给出好或是坏的测试结果（即此结果与被测试芯片实际的好坏无关）。

给出所有芯片的测试结果，问哪些芯片是好芯片。

输入格式

输入数据第一行为一个整数 n，表示芯片个数。

第二行到第 n+1 行为 n*n 的一张表，每行 n 个数据。表中的每个数据为 0 或 1，在这

n 行中的第 i 行第 j 列（1≤i, j≤n）的数据表示用第 i 块芯片测试第 j 块芯片时得到的测试结果，1 表示好，0 表示坏，i=j 时一律为 1（并不表示该芯片对本身的测试结果。芯片不能对本身进行测试）。

输出格式

按从小到大的顺序输出所有好芯片的编号

样例输入

3

1 0 1

0 1 0

1 0 1

样例输出

1 3

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main() {
    int i, j, k, l, n, a[20][20], b[20] = {0}, c[20] = {0};
    scanf("%d", &n);
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &a[i][j]);
    l = 1;
    for (i = 0; i < n; i++) {
        if (b[i] != 0)
            continue;
        b[i] = l++;
        for (j = i + 1; j < n; j++) {
            if (b[j] != 0)
                continue;
            for (k = 0; k < n; k++)
                if (a[i][k] != a[j][k])
                    break;
            if (k == n)
                b[j] = b[i];
        }
    }
```

```
    int max = 0;
    for (i = 0; i < n; i++)
        c[b[i]]++;
    for (i = 0; i < l; i++)
        if (c[max] < c[i])
            max = i;
    for (i = 0; i < n; i++)
        if (max == b[i])
            printf("%d ", i + 1);

    return 0;
}
```

5.1.9 龟兔赛跑预测（数组模拟）

话说这个世界上有各种各样的兔子和乌龟，但是研究发现，所有的兔子和乌龟都有一个共同的特点——喜欢赛跑。于是世界上各个角落都不断在发生着乌龟和兔子的比赛，小华对此很感兴趣，于是决定研究不同兔子和乌龟的赛跑。他发现，兔子虽然跑比乌龟快，但它们有众所周知的毛病——骄傲且懒惰，于是在与乌龟的比赛中，一旦任一秒结束后兔子发现自己领先 t 米或以上，它们就会停下来休息 s 秒。对于不同的兔子，t，s 的数值是不同的，但是所有的乌龟却是一致——它们不到终点决不停止。

然而有些比赛相当漫长，全程观看会耗费大量时间，而小华发现只要在每场比赛开始后记录下兔子和乌龟的数据——兔子的速度 v1（表示每秒兔子能跑 v1 米），乌龟的速度 v2，以及兔子对应的 t，s 值，以及赛道的长度 l——就能预测出比赛的结果。但是小华很懒，不想通过手工计算推测出比赛的结果，于是他找到了你——清华大学计算机系的高才生——请求帮助，请你写一个程序，对于输入的一场比赛的数据 v1，v2，t，s，l，预测该场比赛的结果。

输入格式

输入只有一行，包含用空格隔开的五个正整数 v1，v2，t，s，l，其中(v1,v2<=100;t<=300;s<=10;l<=10000 且为 v1,v2 的公倍数)

输出格式

输出包含两行，第一行输出比赛结果——一个大写字母“T”或“R”或“D”，分别表示乌龟获胜，兔子获胜，或者两者同时到达终点。

第二行输出一个正整数，表示获胜者(或者双方同时)到达终点所耗费的时间(秒数)。

样例输入

10 5 5 2 20

样例输出

D
4
样例输入
10 5 5 1 20
样例输出
R
3
样例输入
10 5 5 3 20
样例输出
T
4

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main() {
    int v1, v2, t, s, l;
    cin >> v1 >> v2 >> t >> s >> l;
    int i, t1 = 0, t2 = l / v2;
    int s1 = 0, s2 = 0;
    for (i = 1; i <= t2; i++) {
        s2 += v2;
        if (t1 < i) {
            t1 = i;
            s1 += v1;
        }
        if (s1 == l)
            break;
        if (t1 == i && s1 - s2 >= t)
            t1 += s;
    }
    if (s1 == l && s2 == l)
        cout << "D" << endl << t2;
    else if (s1 == l)
        cout << "R" << endl << t1;
    else
        cout << "T" << endl << t2;
```

```
    return 0;
}
```

5.1.10 报时助手（字符串条件判断）

给定当前的时间，请用英文的读法将它读出来。

时间用时 h 和分 m 表示，在英文的读法中，读一个时间的方法是：

如果 m 为 0，则将时读出来，然后加上“o'clock”，如 3:00 读作“three o'clock”。

如果 m 不为 0，则将时读出来，然后将分读出来，如 5:30 读作“five thirty”。

时和分的读法使用的是英文数字的读法，其中 0~20 读作：

0:zero, 1: one, 2:two, 3:three, 4:four, 5:five, 6:six, 7:seven, 8:eight, 9:nine, 10:ten, 11:eleven, 12:twelve, 13:thirteen, 14:fourteen, 15:fifteen, 16:sixteen, 17:seventeen, 18:eighteen, 19:nineteen, 20:twenty。

30 读作 thirty，40 读作 forty，50 读作 fifty。

对于大于 20 小于 60 的数字，首先读整十的数，然后再加上个位数。如 31 首先读 30 再加 1 的读法，读作“thirty one”。

按上面的规则 21:54 读作“twenty one fifty four”，9:07 读作“nine seven”，0:15 读作“zero fifteen”。

输入格式

输入包含两个非负整数 h 和 m，表示时间的时和分。非零的数字前没有前导 0。h 小于 24，m 小于 60。

输出格式

输出时间时刻的英文。

样例输入

```
0 15
```

样例输出

```
zero fifteen
```

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main() {
    int h, m;
    string en0to19[] = {
        "zero", "one", "two", "three", "four", "five",
        "six", "seven", "eight", "nine", "ten", "eleven",
        "twelve", "thirteen", "fourteen", "fifteen",
```

```
        "sixteen", "seventeen", "eighteen", "nineteen"
    };
    string en20to50[] = {
        "twenty", "thirty", "forty", "fifty"
    };
    string  oclick = "o'clock";
    scanf("%d%d", &h, &m);
    if (h >= 20) {
        printf("%s", en20to50[0]);
        if (h % 20 > 0)
            printf(" %s", en0to19[h % 20]);
    } else {
        printf("%s", en0to19[h]);
    }
    if (m == 0) {
        printf(" %s", oclick);
    } else if (m >= 20) {
        printf(" %s", en20to50[m / 10 - 2]);
        if (m % 10 > 0)
            printf(" %s", en0to19[m % 10]);
    }//printf("%s",a)会将变量 a 作为字符串类型进行格式化输出
    else {
        printf(" %s", en0to19[m]);
    }
    return 0;
}
```

5.1.11 Huffuman 树（貪心 Huffuman）

Huffman 树在编码中有着广泛的应用。在这里，我们只关心 Huffman 树的构造过程。给出一列数{pi}={p0, p1, …, pn-1}，用这列数构造 Huffman 树的过程如下：

1. 找到{pi}中最小的两个数，设为 pa 和 pb，将 pa 和 pb 从{pi}中删除掉，然后将它们的和加入到{pi}中。这个过程的费用记为 pa + pb。

2. 重复步骤 1，直到{pi}中只剩下一个数。

在上面的操作过程中，把所有的费用相加，就得到了构造 Huffman 树的总费用。

本题任务：对于给定的一个数列，现在请你求出用该数列构造 Huffman 树的总费用。

例如，对于数列{pi}={5, 3, 8, 2, 9}，Huffman 树的构造过程如下：

1. 找到{5, 3, 8, 2, 9}中最小的两个数，分别是 2 和 3，从{pi}中删除它们并将和 5 加入，得到{5, 8, 9, 5}，费用为 5。

2. 找到{5, 8, 9, 5}中最小的两个数，分别是 5 和 5，从{pi}中删除它们并将和 10 加入，得到{8, 9, 10}，费用为 10。

3. 找到{8, 9, 10}中最小的两个数，分别是 8 和 9，从{pi}中删除它们并将和 17 加入，得到{10, 17}，费用为 17。

4. 找到{10, 17}中最小的两个数，分别是 10 和 17，从{pi}中删除它们并将和 27 加入，得到{27}，费用为 27。

5. 现在，数列中只剩下一个数 27，构造过程结束，总费用为 5+10+17+27=59。

输入格式

输入的第一行包含一个正整数 n（n<=100）。

接下来是 n 个正整数，表示 p0, p1, …, pn-1，每个数不超过 1000。

输出格式

输出用这些数构造 Huffman 树的总费用。

样例输入

5

5 3 8 2 9

样例输出

59

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main() {
    int i, j, k, l, n;
    int a[100];
    int count = 0;
    cin >> n;
    for (i = 0; i < n; i++)
        cin >> a[i];
    for (i = n; i > 1; i--) {
        k = 0, l = 1;
        if (a[0] < a[1])
            k = 1, l = 0;
        for (j = 2; j < i; j++) {
            if (a[j] <= a[k]) {
                if (a[j] < a[l])
                    k = l, l = j;
```

```
            else
                k = j;
        }
    }
    a[l] += a[k];
    count += a[l];
    for (j = k; j < i - 1; j++)
        a[j] = a[j + 1];
}
cout << count;
return 0;
}
```

5.1.12 高精度加法（数组高精度）

输入两个整数 a 和 b，输出这两个整数的和。a 和 b 都不超过 100 位。

由于 a 和 b 都比较大，所以不能直接使用语言中的标准数据类型来存储。对于这种问题，一般使用数组来处理。

定义一个数组 A，A[0]用于存储 a 的个位，A[1]用于存储 a 的十位，依此类推。同样可以用一个数组 B 来存储 b。

计算 c = a + b 的时候，首先将 A[0]与 B[0]相加，如果有进位产生，则把进位（即和的十位数）存入 r，把和的个位数存入 C[0]，即 C[0]等于(A[0]+B[0])%10。然后计算 A[1]与 B[1]相加，这时还应将低位进上来的值 r 也加起来，即 C[1]应该是 A[1]、B[1]和 r 三个数的和. 如果又有进位产生，则仍可将新的进位存入到 r 中，和的个位存到 C[1]中。依此类推，即可求出 C 的所有位。

最后将 C 输出即可。

输入格式

输入包括两行，第一行为一个非负整数 a，第二行为一个非负整数 b。两个整数都不超过 100 位，两数的最高位都不是 0。

输出格式

输出一行，表示 a + b 的值。

样例输入

```
20100122201001221234567890
2010012220100122
```

样例输出

```
20100122203011233454668012
```

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

```
#include <iostream>
#include <cstring> //strlen( )取字符串长度
using namespace std;
int main()
{
    int i;
    char a[101], b[101], c[102] = {0};
    cin>>a>>b;
    int alen = strlen(a);
    int blen = strlen(b);
    for (i = 0; i < alen || i < blen; i++)
    {
        if (i < alen)
            c[i] += a[alen-i-1] - '0';
        if (i < blen)
            c[i] += b[blen-i-1] - '0';
        if (c[i] >= 10)
        {
            c[i+1] = c[i] / 10;
            c[i] %= 10;
        }
    }
    if (alen < blen) alen = blen;
    if (c[alen] > 0) cout<<int(c[alen]); //printf("%d", c[alen]);
    for (i = alen - 1; i >= 0; i--)
       // printf("%d", c[i]); //将字符型转为数值型输出
       cout<<int(c[i]);
    return 0;
}
```

5.1.13 阶乘计算（高精度 较简代码）

输入一个正整数 n，输出 n!的值。

其中 n!=1*2*3*…*n。

n!可能很大，而计算机能表示的整数范围有限，需要使用高精度计算的方法。使用一个数组 A 来表示一个大整数 a，A[0]表示 a 的个位，A[1]表示 a 的十位，依次类推。

将 a 乘以一个整数 k 变为将数组 A 的每一个元素都乘以 k，请注意处理相应的进位。

首先将 a 设为 1，然后乘 2，乘 3，当乘到 n 时，即得到了 n!的值。

输入格式：输入包含一个正整数 n，n<=1000。

输出格式：输出 n!的准确值。

样例输入

10

样例输出

3628800

```
#include<iostream>
using namespace std;
int main( ) {
    int n, s, i, j, t, d = 1; //d 阶乘的位数
    int a[10000] = {};
    cin >> n;
    a[0] = 1;//第一个元素是 1，表示 1 的阶乘
    for (i = 2; i <= n; i++) {
        s = 0;
        for (j = 0; j < d; j++) {
            t = a[j] * i + s; //当前位置的数=之前在这个位置上的数乘以 i，然后加上前一位数的进位
            a[j] = t % 10;    //该位置的数 即为当前值对 10 的取余
            s = t / 10;   //进位即为该位置的数 除以 10
        }
    while (s!=0) {    //如果还不是 0 说明还要向上一位进位
            a[d++] = s % 10;
            s = s / 10;
        }
    }
    for (i = d-1; i >= 0; i--)  //d-1 去掉前导零
        cout<<a[i];
    return 0; }
```

蓝桥杯大赛青少组 C++组赛前集训包

6 第 01 课 基本数据类型及运算符

6.1.1、基本数据类型及类型转换

1. 基本数据类型

整型：int、longlong

int 与 long long 的存储范围不同，一般情况下我们用 int 即可，但是如果题目中给的数据范围较大，则选择使用 long long。

布尔型：bool

布尔类型只有两个值，false 和 true。通常用来判断条件是否成立。 字符型：char

char 类型的变量代表的是单个字符，

例如：

```
char a;//定义一个字符型变量
cin>>a;//从键盘输入一个字符存入变量 a 中。
a=‘*’; //给变量 a 赋一个字符’*’。
```

注意：字符是用单引号来表示’’。

实型：float、double

float 与 double 的精确度不同，如果题目没有特别要求，我们一般使用 double， 如果题目明确告诉你使用单精度浮点数数据类型或者 float，则要使用 float。

2. 数据类型的转换

（1）自动类型转换（隐式类型转换）

在不同数据类型的混合运算中，编译器会隐式地进行数据类型转换，称为自动类型转换。自动类型转换遵循下面的规则：

①若参与运算的数据类型不同，则先转换成同一类型，然后进行运算。

②转换按数据长度增加的方向进行，以保证精度不降低。例如 int 类型和 long 类型运算时，先把 int 类型转成 long 类型后再进行运算。

③在赋值运算中，赋值号两边的数据类型不相同时，将把右边表达式值的类型转换为左边变量的类型。如果右边表达式值的数据类型长度比左边长时，将丢失一部分数据。

④在赋值语句中，赋值号两边数据类型一定是相兼容的类型，如果等号两边数据类型不兼容，语句在编译时会报错。

（2）强制类型转换（显示类型转换）

自动类型转换不能实现目的时，可以显式进行类型转换，称为强制类型转换。一般形式：（数据类型）（表达式）

注意：数据类型加小括号这个整体才是强制类型转换的符号， 后面的表达式可以 加小括号，也可以不加，如果不加的话则遵循就近原则，谁离得强制类型转换符 号近，系统则强制类型转换谁。

例： int a;a=(int)1.5;a 的值为 1。

6.21.2、变量与常量

1. 变量

变量可以看作是存储数据的小盒子，一个小盒子里面只能存放一个具体的值，而 且这个值可以改变。

例如：inta= 3；a 就是一个变量

2. 常量

常量是固定值，在程序执行期间不会改变。这些固定的值，又叫做字面量。

常量可以是任何的基本数据类型，可分为整型数字、浮点数字、字符、字符串和布尔值。

常量就像是常规的变量，只不过常量的值在定义后不能进行修改。

6.31.3、字符与字符串

1、字符就是单个字符， 字符串就是多个字符的集合。

2、单个空白字符和空白字符串是两个概念，在 C++中字符就是单个字符，字符 串是用\0 结尾的，字符和字符串在操作上也不同，复制等等是不一样的。

字符串简介：字符串或串(String)是由数字、字母、下划线组成的一串字符。一般记为 s=″a1a2・・・an″(n>=0)。它是编程语言中表示文本的数据类型。

在程序设计中，字符串(string)为符号或数值的一个连续序列，如符号串(一串字符)或二进制数字 串(一串二进制数字)。

6.41.4、运算符

1. 赋值运算符

在 C++里面，一个等号“=”代表的是赋值，赋值运算符是将赋值运算符右侧的值赋值给左侧的变量。

2. 算术运算符

C++中有 5 个基本的算术运算符： +（加） -（减） 、*（乘） 、/（除） 、%（取余）。

注意：

① 两个整数相除，得到的结果值保留整数位

② 取余运算符两边的数字必须都得是整数

3. 逻辑运算符

C++中有三个基本的逻辑运算符： &&（与） 、||（或） 、!（非）

逻辑非： 经过逻辑非运算，其结果与原来相反。

逻辑与： 若参加运算的某个条件不成立，其结果就不成立，只有当参加运算的所 有条件都成立，其结果才成立。

逻辑或： 若参加运算的某个条件成立，其结果就成立，只有当参加运算的所有条 件都不成立，其结果才不成立。

4. 关系运算符

C++中有六个基本的关系运算符： >（大于） 、>=（大于等于） 、<（小于） 、<= （小于等于） 、 ==（等于） 、 !=（不等于）

7 第 02 课 基本程序结构

7.12.1、顺序结构程序设计

1. 输入语句：

cin 是 C++的输入语句,与 cout 语句一样,为了叙述方便,常常把由 cin 和运 算符">>"实现输入的语句称为输入语句或 cin 语句。

cin 语句的一般格式为：

cin>>变量 1>>变量 2>>……>>变量 n;

与 cout 类似，一个 cin 语句可以分写成若干行，如：

```
cin>>a>>b>>c>>d;
```

也可以写成

```
cin>>a;
cin>>b;
cin>>c;
cin>>d;
```

以上书写变量值均可以以从键盘输入：1 2 3 4

也可以分多行输入数据：

```
1
2 3
4
```

2. 输出语句：

cout 语句一般格式为：

cout<<项目 1<<项目 2<<…<<项目 n;

功能：

（1） 如果项目是表达式，则输出表达式的值。

（2） 如果项目加引号，则输出引号内的内容。

输出总结：

cout<<项目 1<<项目 2<<…<<项目 n;

①输出语句可以输出多项内容，用<<连接;

②如果输出项目是"2+3"，则输出 2+3;

③如果输出项目是 2+3，则输出 5;

④如果输出遇到 endl，则换行。

3. 输出格式控制：

7.22.2、分支结构程序设计

7.2.1 1. if-else 语句

一般形式：
```
if（表达式）
  {
  语句块 1;
  }
else
  {
  语句块 2;
  }
```
判断表达式的逻辑值，当表达式的值非 0，则执行语句块 1，当表达式的值为 0 的时候，执行语句块 2.

7.2.2 2. switch 语句

switch 语句是多分支选择语句，也叫开关语句。switch 语句基本格式及框架图如下：
```
switch（表达式）{
case 常量表达式 1：[语句组 1] [break;]
case 常量表达式 2：[语句组 2] [break;]
……
case 常量表达式 n：[语句组 n] [break;]
[default:语句组 n+1]
}
```
功能：首先计算表达式的值，case 后面的常量表达式值逐一与之匹配，当某一个 case 分支中的常量表达式值与之匹配时，则执行该分支后面的语句组，然后 顺序执行之后的所有语句，直到遇到 break 语句或 switch 语句的右括号“}”为止。如果 switch 语句中包含 default，default 表示表达式与各分支常量表达式的值都不匹配时，执行其后面的语句组，通常将 default 放在最后。

规则：

（1）合法的 switch 语句中的表达式，其取值只能是整型、字符型、布尔型或者枚举型

（2）常量表达式是由常量组成的表达式，值的类型与表达式类型相同。

（3）任意两个 case 后面的常量表达式值必须各不相同，否则将引起歧义

（4）“语句组”可以使一个语句也可以是一组语句

（5）基本格式中的[ ]表示可选项

7.2.3 3. 分支语句嵌套

在 if 语句中又包含一个或多个 if 语句称为 if 语句的嵌套。

一般形式如下：
```
if( )
if( ) 语句 1
      else 语句 2
```

```
else
   if( )语句 3
else 语句 4
```
应当注意 if 与 else 的配对关系。else 总是与它上面最近的、且未配对的 if 配对。如果 if 与 else 的数目不一样，为实现程序设计者的企图，可以加花括号来确定配对关系。例如：
```
if( )
{
  if ( ) 语句 1
} //这个语句是上一行 if 语句的内嵌 if
else 语句 2//本行与第一个 if 配对
```
这时{ }限定了内嵌 if 语句的范围，{ }外的 else 不会与{ }内的 if 配对。关系清楚，不易出错。

7.32.3、循环结构程序设计

1、while 语句

while 死循环结构:
（1）格式：
```
while(1) {
   循环语句;
}
```
（2）功能
不断地执行循环体中的语句。
（1）格式
格式 1：
```
while(表达式)
   语句;
```
格式 2：
```
while(表达式){
语句 1;
语句 2;
……
}
```
（2）功能
当表达式的值非 0 时，不断地执行循环体中的语句。所以，用 while 语句实现的循环被称为“当型循环”。

7.3.1 2. for 语句

for 循环格式
格式 1:
```
for(循环变量初始化;循环条件;循环变量增量)
语句;
```

格式 2:
for(循环变量初始化;循环条件;循环变量增量)
 { 语句 1;
    语句 2;

……

}

7.3.2 3. do-while 语句

do{
    语句;
}while(表达式); //尤其注意圆括号后面的分号 ；
do-while 循环与 while 循环的不同在于：
    它先执行循环中的语句，然后再判断表达式是否为真； 如果为真则继续循环，如果为假，则终止循环。因此，do-while 循环至少要执行一次循环语句。

7.3.3 4、循环结构嵌套 循环的嵌套：

（1） 一个循环体内包含着另一个完整的循环结构，就称为嵌套循环。
（2） 内嵌的循环又可以嵌套循环，从而构成多重循环。
（3） 三种循环可以相互嵌套。
注意：
①嵌套的循环控制变量不应同名，以免造成混乱。
②内循环变化快，外循环变化慢。
③正确确定循环体。
④循环控制变量与求解的问题挂钩。

7.3.4 5. break 语句

中断所在循环体，跳出本层循环。

8 第 03 课 数组

8.1 3.1、一维数组及二维数组

8.1.1 1. 一维数组

    申请 10 个整数数据类型的变量可以这么写：int a[10];
int a[10];这行语句代表同时定义了 10 个整型变量，就如同 10 个“小房子”并排放在

了一起。
那么我们如何使用这些变量呢？
首先，[ ]里的数字表示需要定义的变量的个数，我们这里定义了 10 个。这 10 个变量分别用 a[0]、a[1]、a[2]、a[3]、a[4]、a[5]、a[6]、a[7]、a[8]、a[9]来 表示。
注意：我们要表达数组中某一个元素的格式是：数组名[下标]。在 C++中，下标是从 0 开始的，所以一个大小为 n 的数组，它的有效下标是 0~n-1。0 是下标，a[0]用来存值
数组：由具有相同数据类型的固定数量的元素组成的结构。
例如：int a[10];
double b[10], c[5];
注意：数组定义时的一个关键点是数组的长度如何选择。
数组元素的引用：
(1)下标可以是整型常量或整型表达式;
a[3]=3;
或: int i=3;
a[i]=3;
(2)下标在 0~4 之内, 即 a[0]~a[4]
注意：下标不要超范围
(3)可以单独针对每个元素赋值，
如：a[4] = 5;
也可以这么用：
int i = 4;
a[i] = 5;
(4) 每个元素都是一个变量，数组是“一组变量”，而不是一个变量。

8.1.2 2. 二维数组

二维数组定义的一般格式：
类型名 数组名[常量表达式 1][常量表达式 2];
通常二维数组中的第一维表示行下标，第二维表示列下标。行下标和列下标都是从 0 开始的。例如：
int num[4][6];
二维数组的使用与一维数组类似，引用的格式为：
数组名[下标 1][下标 2]

使用数组时特别注意下标不能越界。
使用二维数组时，需要区分是处理行数据、列数据，还是处理所有数据的行列下 标。
遍历一个二维数组要使用二重循环。

8.2 3.2、数组的输入和输出

8.2.1 1. 一维数组的输入输出

利用一层 for 循环实现控制下标的变化从而实现对一维数组元素的输入以及输

出。
例如：for(int i=0; i<5; ++i){
    cin>> a[i];
    }
利用循环实现了对一维数组元素的输入。

8.2.2 2. 二维数组的输入输出

二维数组的输入方式有两种：
（1）按行输入：
输入 n 行 m 列数据(n、m 均小于 100)：
```
int n, m, a[105][105];
cin>> n >> m;
for(int i=1; i<=n; i++){ //行数变化
for(int j=1; j<=m; j++){ //列数变化
cin>> a[i][j];          //按行输入
}
}
```
（2）按列输入：
输入 n 行 m 列数据(n、m 均小于 100) ：
```
int n, m;
cin >> n >> m;
for (int j = 1; j <= m; j++) {
    for (int i = 1; i <= n; i++) {
        cin >> a[i][j];
    }
}
```
二维数组的输出我们只需要掌握住按行输出即可：
```
for(int i=1;i<=n;i++){//控制行数
    for(int j=1;j<=m;j++){
        cout<<a[i][j]<<" ";
}
cout<<endl;
}
```

8.33.3、数组元素的遍历

8.3.1 1. 一维数组的遍历

将存放在一维数组中的元素依次查看一遍并寻找符合条件的数的过程就是对一 维数组的遍历。

8.3.2 2. 二维数组的遍历

二维数组遍历和一维数组遍历类似，只不过在遍历到一维元素时，由于元素是一 维数组还需要遍历，构成双重循环。使用双重循环遍历二维数组时，外层循环的 次数使用数组元素的行数来进行控制，内层循环的次数是使用每个一维数组的元 素的，也就是二维数组的列数来进行控制。

8.4 3. 4、数组元素排序

8.4.1 1. 选择排序

选择排序：是一种简单直观的排序算法。它的工作原理是每一次从待排序的数据 元素中选出最小（或最大）的一个元素，存放在序列的起始位置，然后，再从剩 余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。以此类 推， 直到全部待排序的数据元素排完。
例如，6 个数按照从小到大排序：

```
#include <iostream>
using namespace std;
int main( ) {
    int a[6], i, j, t;
    for ( i = 0 ; i < 6 ; i++)
        cin >> a[i];
    for ( i = 0 ; i < 5; i++)
        for ( j = i + 1 ; j < 6; j++)
            if ( a[i] > a[j] ) {
                t = a[i] ;
                a[i] = a[j] ;
                a[j] = t ;
            }
    for ( i = 0 ; i < 6 ; i++)
        cout << a[i] << " ";
    return 0;
}
```

可推广到 n 个数的选择排序

8.4.2 2. 冒泡

冒泡排序是一种计算机科学领域的较简单的排序算法。它重复地走访过要排序的元素列，依次比较两个相邻的元素，如果他们的顺序错误就把他们交换过来。走访元素的工作是重复地进行直到没有相邻元素需要交换，也就是说该元素已经排序完成。
这个算法的名字由来是因为越大（小）的元素会经由交换慢慢“浮”到数列的顶端（升序或降序排列），就如同碳酸饮料中二氧化碳的气泡最终会上浮到顶端一样，故名“冒泡排

序”。
例如：5 个数按照从小到大排序：

```
#include <iostream>
using namespace std;
int main( ) {
    int a[5], i, j, t;
    for ( i = 0 ; i < 5 ; i++)
        cin >> a[i];
    for ( i = 0 ; i < 4 ; i++) //比较了四轮
        for ( j = 0 ; j < 4 ; j++) //每轮需要四次比较
            if ( a[j] > a[j + 1] ) {
                t = a[j] ;
                a[j] = a[j + 1] ;
                a[j + 1] = t ;
            }
    for ( i = 0 ; i != 5 ; i++)
        cout << a[i] << ″ ″;
    return 0;
}
```

可推广到 n 个数的冒泡排序。

8.4.3 3. 桶排序

现在如果需要将输入的 5 个数（范围是 0~9）从小到大排序，该怎么办？
例如输入 2 5 2 1 8，则输出 1 2 2 5 8。首先我们先申请一个大小为 10 的数组，int a[10],编号为 a[0]~[9]，并初始化为 0。我们现在只需将 “小房间的值加 1”就可以了，例如： 2 出现了，就将 a[2]的值加 1。
实现过程：

```
#include<iostream>
using namespace std;
int main() {

    int i, j, t, a[10] = {0};  //数组初始化为 0
    for (i = 1; i <= 5; i++) { //循环读入 5 个数
        cin >> t;  //把每一个数读到变量 t 中
        a[t]++;    // t 所对应小房子中的值增加 1
    }
    for (i = 0; i <= 9; i++) { //依次判断 0~9 这个 10 个小房子
        for (j = 1; j <= a[i]; j++) //出现了几次就打印几次
            cout << i << ‘ ‘;
    }
    return 0;
}
```

这种形式的排序就是桶排序，桶排序是要借助于数组来实现的，我们将每个数组元素

看作一个桶，每出现一个数，就在对应编号的桶里面放一个小旗子，最后只要数一数每个桶里面有几个小旗子就 OK 了。

8.53.5、字符数组

8.5.1 1. 一维字符数组

我们将 char 类型定义的数组叫做字符数组，也可以叫它字符串。
字符数组的定义格式如下：
char 数组名[元素个数];
cin 和 scanf 输入方式是没法读入空格，回车，tab 的，如果我们的字符串中需要空格，可以用 gets()来读入一行字符串
例如：

```
char a[10];
gets(a);
cout<<a;
```

注意：gets()函数可以**保留空格，遇到回车结束**，需要一个头文件<cstdio>
字符数组初始化的两种方式：

```
char a[10];
```

（1）char a[10]={‘c’,’o’,’d’。,’u’,’c’,’k’};//一些单个字符
（2）char a[10]={“coduck″}; //一个字符串
输入：

```
char a[10];
```

（1）无空格的字符串输入：
1.利用循环输入：

```
for(int i=0;i<=9;i++){
   cin>>a[i]; //输入固定长度为 10 的字符数组。
  }
```

2.直接 cin ：

```
cin>>a;   //可以输入小于等于 10 的字符数组
```

（2）有空格的字符串输入：
1.**gets(a);** //gets 函数可以保留空格，遇到回车结束！加头文件<cstdio> 说明：遇到没有空格的字符串,可以用 cin>>数组名;来输入。遇到有空格的字 符串,可以用 gets(数组名);来输入。

输出：

```
cout<<a;
```

说明：cout 输出遇到‘\0’结束。
2. 二维字符数组
我们可以借助一个二维数组来存储多个字符串，这个二维字符数组的每一行都是 一个单独的字符串。

9 第 04 课 函数

9.14.1、函数的定义和使用

函数定义

前面我们用过了很多 C++标准函数，但是这些标准函数并不能满足所有需求。当我们需要特定的功能函数时，这就需要我们要学会自定义函数,根据需求定制想要的功能。
函数定义的语法：
返回类型　函数名（参数列表）

```
{
   函数体;
}
```

关于函数定义的几点说明：
（1）自定义函数符合“根据已知计算未知”这一机制， 参数列表相当于已知，是自变量，函数名相当于未知，是因变量。max 函数的功能是找出两个数的最大数，参数列表中 x 相当于已知自变量，max 函数的值 相当于未知因变量。
（2）函数名是标识符， 一个程序中除了主函数名必须为 main 外，其余函数的名字按照标识符的取名规则命名。
（3）参数列表可以是空的，即无参函数，也可以有多个参数，参数之间用逗号隔开，不管有没有参数，函数名后的括号不能省略。参数列表中的每个参数，由参数类型说明和参数名组成。如 max(int a,int b) 函数的参数列表数有两个参数，两个参数类型分别是 int，int，两个参数名分别是 a，b。
（4）函数体是实现函数功能的语句，除了返回类型是 void 的函数，其他函数的函数体中至少有一条语句是“return 表达式;”用来返回函数的值。执行函数过程中碰到 return 语句，将在执行完 return 语句后直接退出函数，不再去执行后面的语句。
（5）返回值的类型一般是前面介绍过的 int、double、char 等类型，也可以是 数组。有时函数不需要返回任何值，例如函数可以只描述一些过程用 printf 向
屏幕输出一些内容，这时只需定义的数返回类型为 void，并且无须使用 return 返回函数的值。

函数使用时，函数名必须与函数定义时完全一致，实参与形参个数相等，类型一致，按顺序一一对应。被调用函数必须是已存在的函数，如果调用库函数，一定要包含相对应的头文件。

9.24.2、函数的递归调用

函数之间有三种调用关系：主函数调用其他函数、其他函数互相调用、函数递归 调用
C++程序从主函数开始执行，主函数由操作系统调用，主函数可以调用其他函　数， 其他函数之间可以互相调用，但不能调用主函数，所有函数是平行的，可以 嵌套调用，但不能嵌套定义。

9.34.3、变量的作用域： 局部变量和全局变量

作用域描述了名称在文件的多大范围内可见可使用。C++程序中的变量按作用域来分，有**全局变量**和**局部变量。**

全局变量：定义在函数外部没有被花括号括起来的变量称为全局变量。 全局变量的作用域是从变量定义的位置开始到文件结束。由于全局变量是在函数
外部定义的，因此对所有函数而言都是外部的，可以在文件中位于全局变量定义后面的任何函数中使用。
局部变量：定义在函数内部作用域为局部的变量局部变量。函数的形参和在该函数里定义的变量都被称为该函数的局部变量。
注意：全局变量和局部变量都有生命周期，变量从被生成到被撤销的这段时间就称为变量的生存期，实际上就是变量占用内存的时间。局部变量的生命周 期是从函数被调用的时刻开始到函数结束返回主函数时结束。而全局变量的生命周期与程序的生命周期是一样的。若程序中全局变量与局部变量同名，且同时有效，则以局部变量优先。即在局部变量的作用范围内，全局变量不起作用。

10 第 05 课 简单算法

10.1 5.1、进制转换

10.1.1 1、r 进制数(非十进制数)转化成十进制数：

各种进位制转换为十进制的方法：分别写出二进制数、八进制数和十六进制数的按权展开式，再按十进制运算规则求和得到的值，即为转换后的十进制 数。

10.1.2 2、十进制数转化成 r 进制数：

分整数和小数两部分分别转换处理，最后再求和。
整数部分的转换方法是：
不断的除以 r 取余数，直到商为 0，余数从下到上排列（除 r 取余,逆序排列）；
小数部分的转换方法是：
不断乘以 r 取整数，整数从上到下排列（乘 r 取整,顺序排列）。

3、二进制、八进制、十六进制数间的转换：

二进制、八进制、十六进制之间的对应规则如下：
每 3 位二进制对应 1 位八进制数；每 4 位二进制对应 1 位十六进制数。
（1）二进制转化成八(十六)进制：分为如下三个步骤
整数部分：小数点为基准从右向左按三(四)位进行分组。
小数部分：小数点为基准从左向右按三(四)位进行分组，不足补零。
（2） 八（十六）进制转换为二进制
八进制数转换成二进制数： 只需将 1 位八进制数转为 3 位二进制数。
十六进制数转换成二进制数： 只需将 1 位十六进制数转为 4 位二进制数。

10.2 5.2、模拟算法

在我们所解决的题目中，有一类问题是模拟一个游戏的对弈过程，或者模拟一项任务的操作过程，进行统计计分，判断输赢等。这些问题很难建立数学模型用特定算法解决，一般只能采用“模拟”法。

用模拟法解决必须关注以下几个问题：

审题要仔细，游戏规则不能错；分析要全面，各种特例不能丢； 编程要细心，测试运行要到位。

例如：

有一天，一只蚱蜢像往常一样在草地上愉快地跳跃，它发现了一条写满了英文字 母的纸带。蚱蜢只能在元音字母（A、E、I、O、U） 间跳跃，一次跳跃所需的能 力是两个位置的差。纸带所需的能力值为蚱蜢从纸带开头的前一位置根据规则调 到纸带结尾的后一个位置的过程中能力的最大值。

蚱蜢想知道跳跃纸带所需能力值（最小） 是多少。

【输入格式】

一行一个字符串，字符串场不超过 100.

【输出格式】

一行一个整数，代表（最小）能力值。

【输入样例】

KMLPTGFHNBVCDRFGHNMBVXWSQFDCVBNHTJKLPMNFVCKMLPTGFH

NBVCDRFGHNMBVXWSQFDCVBNHTJKLPMNFVC

【输出样例】

85

【问题分析】

从头到尾枚举纸带的每一个字母，按照规则模拟蚱蜢在元音字母间跳跃的过程，

打擂台记录能力值。

【示例代码】

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    //freopen("grasshopper.in", "r", stdin); //文件读写
    //freopen("grasshopper.out", "w", stdout);
    string a;
    cin >> a;
    int ans = 0, pre = 0;
    for (int i = 0; i < a.length(); i++) {
        if (a[i] == 'A' || a[i] == 'E' || a[i] == 'I' || a[i] == 'O' || a[i] ==
'U' || a[i] == 'Y') {
            if (ans <= (i + 1 - pre)) {
                ans = i + 1 - pre;
            }
            pre = i + 1;
        }
    }
    if (ans <= a.length() + 1 - pre) {
        ans = a.length() + 1 - pre;
    }
    cout << ans << endl;
```

```
    return 0;
}
```

10.3 5.3、枚举算法

计算机的特点之一就是运算速度快，善于重复做一件事。“穷举”正是基于这一特点的最古老的算法。它一般是在一时找不出解决问题的更好途径，即从数学上找不到求解的公式或规则时，根据问题中的“约束条件”，将解的所有可能情况一一列举出来，然后再逐个验证是否符合整个问题的求解要求，从而求得问题的可行解或者最优解。

例题：火柴棒等式

【问题描述】

给出 n 根火柴棒， 可以拼出多少个形如“A+B=C”的等式？等式中的 A、B、C 是用火柴棒拼出的整数（若该数非零，则最高位不能是 0） 。 用火柴棒拼数字 0~9 的拼法如下图所示。

需要注意以下几点：

（1） 加号与等号各自需要两根火柴棒。

（2） 如果 A ≠ B，则 A+B=C 与 B+A=C 视为不同的等式（A、B、C 均大 于或等于 0）。

（3） n 根火柴棒必须全部用上（n≤24）。

【输入样例】

14

【输出样例】

2

【样例说明】

两个等式分别为：0+1=1 和 1+0=1。

【问题分析】

首先，预处理每个数字（0~9）需要用几根火柴棒， 存储在数组 f 中。然后穷举 a 和 b，算出它们的和 c，再判断下列约束条件是否成立：

f （a）+ f （b）+f （c）= n-4。

现在的问题是： a 和 b 的范围有多大？可以发现尽量用数字 1 拼成的数比较大，分析可知最多不会超过 1111。程序实现时，分别用三个循 环语句预处理好所有两位数、三位数、四位数构成所需要的火柴棒数量。

【示例代码】

```
#include<stdio.h>
int fun(int x) { //用来计算一个数所需要用火柴棍的总数
    int num = 0; //用来计数的变量，所以需要初始化
    int f[10] = {6, 2, 5, 5, 4, 5, 6, 3, 7, 6}; //用一个数组来记录 0~9 每个数字需要用多少根火柴棍
    while (x / 10 != 0) { //如果 x/10 的商不等于 0，说明这个数至少有两位
```

```
        num += f[x % 10]; //获得 x 的末尾数字并将此数所需要用到的火柴棍根数累加到 num 中
        x = x / 10; //去掉 x 的末尾数字，例如 x 的值为 123 则现在 x 的值为 12
    }
    num += f[x];  //最后加上此时 x 所需用到的火柴棍的根数（此时 x 一定是一位数）
    return num;  //返回需要火柴棍的总根数
}
int main() {
    int a, b, c, m, sum = 0; //sum 用来计数，因此要初始化
    scanf("%d", &m); //读入火柴棍的个数
    /*开始枚举 a 和 b*/
    for (a = 0; a <= 1111; a++) {
        for (b = 0; b <= 1111; b++) {
            c = a + b; //计算出 c
            if (fun(a) + fun(b) + fun(c) == m - 4) {
                printf("%d+%d=%d\n", a, b, c);
                sum++;
            }
        }
    }
    printf("一共可以拼出%d 个不同的等式", sum);
    return 0;
}
```

11 第 06 课 基本数据结构

11.1 6.1、结构体

11.1.1 1、结构体的定义：

在存储和处理大批量数据时，一般会使用数组来实现，但是每一个数据的类型及含义必须一样。如果需要把不同类型、不同含义的数据当作一个整体来 处理，比如 1000 个学生的姓名、性别、年龄、体重、成绩等，怎么办？ C++提供了结构体。
C++中的结构体是由一系列具有相同类型或不同数据类型的数据构成的数据集合。使用结构体，必须要先声明一个结构体类型，再定义和使用结构体变量。 结构体类型的声明格式如下：

```
struct   类型名{ 数据类型 1   数据类型 2
                 成员名 1;  成员名 2;…
};
```

也可以把结构体类型声明和变量定义在一起，格式如下：

```
struct   类型名{
数据类型 1 数据类型 2
…  成员名 1;  成员名 2;
}变量名;
```

11.1.2 2、结构体变量的使用：

结构体变量具有以下特点：
（1）可以对结构体变量的整体进行操作。例如，swap（a[i],a[j]）。
（2）可以对结构体变量的成员进行操作。
引用结构体变量中的成员的格式为：结构体变量名.成员名
（3）结构体变量的初始化方法与数组类似。
student s={"xiaomming",'f',16,169};

11.2 6.2、栈

11.2.1 1、栈的基本概念：

栈（Stack）是限制在表的一端进行插入和删除操作的线性表
后进先出 LIFO（Last In First Out）
先进后出 FILO（First In Last Out）
栈顶（Top）：允许进行插入、删除操作的一端，又称表尾。用栈顶指针（top）来指示栈顶元素。
空栈：当表中没有元素时称为空栈。

11.2.2 2、栈的顺序表示与实现：

栈的顺序存储结构简称为顺序栈，和线性表类似，用一维数组来存储栈。

11.2.3 3、栈的应用

（1）括号匹配检查
【题目描述】
假设表达式中允许包含圆括号和方括号两种括号，其嵌套的顺序随意，如([]())
或[([][])]等为正确的匹配，[(])或([]()或(()))均为错误的匹配。本题的任务是检验一个给定的表达式中的括号是否匹配正确 。
输入一个只包含圆括号和方括号的字符串，判断字符串中的括号是否匹配， 匹配就输出“OK”，不匹配就输出“Wrong”。
【输入】
一行字符，只含有圆括号和方括号，个数小于 255
【输出】
匹配就输出一行文本“OK“，不匹配就输出一行文本”Wrong”
【样例输入】
[(])
【样例输出】
Wrong
【解决思路】

使用栈来实现，首先输入字符串，遍历字符串：如果是左括号，进栈；
如果是右括号，跟栈顶数据比较
如果和栈顶的左括号匹配，出栈
如果不匹配，输出“Wrong”
遍历结束后，栈中还有括号，输出“Wrong”；没有，输出“Ok”
【代码示例】

```
#include<iostream>
#include<string>
using namespace std;
int main() {
    char a[256];
    string s;
    int i, top;
    cin >> s;
    top = 0;
    for (i = 0; i < s.size(); i++) {
        if (s[i] == '(' || s[i] == '[') {
            a[++top] = s[i];;

        }
        if (s[i] == ')') {
            if (a[top] == '(') top--;
            else {
                top++;
            }
        }
        if (s[i] == ']') {
            if (a[top] == '[') top--;
            else

            {
                top++;
            }
        }
    }
    if (top == 0) cout << "OK" << endl;
    else cout << "Wrong" << endl;
    return 0;
}
```

（2）铁轨问题
代码省略

11.3 6.3、队列

11.3.1 1、队列定义

队列是一种先进先出（FIFO）的线性表。只能在线性表的一端进行插入操作，在另一端进行删除操作。类似与生活中的排队购票，先来先买，后来后买。
在不断入队、出队的过程中，队列将会呈现初以下几种状态：
队满：队列空间已被全部占用。
队空：队列中没有任何元素。
溢出：当队列满时，却还有元素要入队，就会出现“上溢”；当队列已空， 却还要做“出队”操作，就会出现“下溢”。两种情况合在一起称为队列的“**溢出**”。

11.3.2 2、队列的应用

【题目描述】
假设在周末舞会上，男士们和女士们进入舞厅时，各自排成一队。跳舞开始时， 依次从男队和女队的队头上各出一人配成舞伴。规定每个舞曲能有一对跳舞者。 若两队初始人数不相同，则较长的那一队中未配对者等待下一轮舞曲。现要求写
一个程序，模拟上述舞伴配对问题。(0<m,n,k<1000)
【输入】
第一行男士人数 m 和女士人数 n;
第二行舞曲的数目 k。
【输出】
共 k 行，每行两个数，表示配对舞伴的序号，男士在前，女士在后。
【样例输入】
2 4
6
【样例输出】
1 1
2 2
1 3
2 4
1 1
2 2
【解题思路】
显然，舞伴配对的顺序符合“先进先出”，所以用两个队列分别存放男士队伍和 女士队伍，然后模拟 k 次配对：每次取各队队头元素“配对”输出，并进行“出
队”和重新“入队”的操作。
【代码示例】

```
#include<iostream>
using namespace std;
int main() {
    int a[10001], b[10001], f1 = 1, f2 = 1, r1, r2, k1 = 1;
    int m, n, k;
    cin >> m >> n >> k;
    r1 = m;
    r2 = n;
```

```
    for (int i = 1; i <= m; i++) a[i] = i;
    for (int j = 1; j <= n; j++) b[j] = j;
    while (k1 <= k) {
        cout << a[f1] << " " << b[f1] << endl;
        r1++;
        a[r1] = a[f1];
        f1++;
        r2++;
        b[r2] = b[f2];
        f2++;
        k1++;
    }
    return 0;
}
```

11.4 6.4、树

11.4.1 1、树

（1）树的定义

树(Tree)是 n(n≥0)个结点的有限集,它或为空树(n=0) ;或为非空树,对于非空树 T:有且仅有一个称之为根的结点;
除根结点以外的其余结点可分为 m(m>0)个互不相交的有限集 T1，T2,...Tm, 其中每一个集合本身又是一棵树，并且称为根的子树(SubTree)。

（2）基本术语

根——即根结点(没有前驱)
叶子——即终端结点(没有后继)
森林——指 m 棵不相交的树的集合 (例如删除 A 后的子树个数)
有序树——结点各子树从左至右有序 ,不能互换(左为第一)
无序树——结点各子树可互换位置。双亲——即上层的那个结点(直接前驱)
孩子——即下层结点的子树的根(直接后继)
兄弟——同一双亲下的同层结点(孩子之间互称兄弟)
堂兄弟——即双亲位于同一层的结点 (但并非同一双亲)
祖先——即从根到该结点所经分支的所有结点
子孙——即该结点下层子树中的任一结点
结点——即树的数据元素
结点的度——结点挂接的子树数
结点的层次——从根到该结点的层数 (根结点算第一层)
终端结点——即度为 0 的结点 ,即叶子
分支结点——即度不为 0 的结点(也称为内部结点)
树的度——所有结点度中的最大值
树的深度——指所有结点中最大的层数

（3）树的存储结构

方法 1:数组,称为“父亲表示法”

```
const int m = 10;//树的结点数
struct node
{
    int data, parent;//数据域、指针域
};
node tree[m];
```

优缺点:利用了树中除根结点外每个结点都有唯一的父结点这个性质。很容易找 到树根,但找孩子时需要遍历整个线性表。

方法 2:树型单链表结构,称为“孩子表示法”。

每个结点包括一个数据域和一 个指针域(指向若干子结点)。称为”孩子表示法”。假设树的度为 10 ,树的结点 仅存放字符,则这棵树的数据结构定义如下:

```
const int m = 10; / /树的度
typedef struct node ;
typedef node *tree;
struct node
{
char data;//数据域
tree child[m];//指针域，指向若干孩子结点
}
tree t;
```

缺陷:只能从根(父)结点遍历到子结点,不能从某个子结点返回到它的父结点。但 程序中确实需要从某个结点返回到它的父结点时;就需要在结点中多定义一个指 针变量存放其父结点的信息。这种结构又叫带逆序的树型结构。

方法 3 :树型双链表结构,称为”父亲孩子表示法”。每个结点包括一个数据 域和二个指针域(一个指向若干子结点，一个指向父结点)。假设树的度为 10,树 的结点仅存放字符,则这棵树的数据结构定义如下:

```
const int m= 10; / /树的度
typedef struct node;
typedef node *tree;//声明 tree 是指向 node 的指针类型
struct node
{
char data; // 数据域
tree child[m]; //指针域 ,指向若干孩子结点
tree father ; / /指针域,指向父亲结点

};
tree t;
```

方法 4 :二叉树型表示法,称为”孩子兄弟表示法”。也是一种双链表结构,但每个结点包括一个数据域和二个指针域(一个指向该结点的第一个孩子结点,一 个指向该结点的下一个兄弟结点)。称为”孩子兄弟表示法”。假设树的度为 10 ,
树的结点仅存放字符,则这棵树的数据结构定义如下:

```
typedef struct node; ,
typedef node *tree;
struct node
{
char data; //数据域
```

tree firstchild, next;
//指针域,分别指向第一个孩子结点和下一个兄弟结点}; };
tree t;

（4）树的遍历

在应用树结构解决问题时,往往要求按照某种次序获得树中全部结点的信息,这种 操作叫作树的遍历。遍历的方法有多种,常用的有:

A、**先序(根)遍历**:先访问根结点,再从左到右按照先序思想遍历各棵子树。

B、**后序(根)遍历**:先从左到右遍历各棵子树,再访问根结点。

C、**层次遍历**:按层次从小到大逐个访问,同一层次按照从左到右的次序。

D、**叶结点遍历**:有时把所有的数据信息都存放在叶结点中,而其余结点都是用来表示数据之间的某种分支或层次关系,这种情况就用这种方法。

11.4.2 2、二叉树

（1）二叉树的定义

二叉树(Binary Tree)是 n(n≥0)个结点所构成的集合,它或为空树(n= 0) ;或为非空树,对于非空树 T:

a)有且仅有一个称之为根的结点;

b)除根结点以外的其余结点分为两个互不相交的子集 T 和 T2,分别称为的左子树和右子树，且和 72 本身又都是二叉树。

（2）二叉树的基本特点

a)结点的度小于等于 2

b)有序树（子树有序，不能颠倒）二叉树的五种不同的形态

（3） 二叉树的性质

性质 1： 在二叉树的第 i 层上之多有 2^(i-1)个结点

性质 2： 深度为 k 的二叉树至多有 2^k-1 个结点

性质 3：对于任何一棵二叉树,若 2 度的结点数有 n_2 个,则叶子数 n_0 必定为 n_2+1（即 n_0=n_2+1）

特殊形态的二叉树

a)**满二叉树**： 一颗深度为 k 且有 2^k-1 个结点的二叉树。

b)**完全二叉树**： 深度为 k 的，有 n 个结点的二叉树， 当且仅当其每一个结点都与深度为k 的满二叉树中编号从 1 至 n 的结点一一对应

性质 4： 具有 n 个结点的完全二叉树的深度必为[log_2(n)]+1

性质 5： 对完全二叉树，若从上至下、从左至右编号，则编号为 i 的结点，其左孩子编号必为 2i，其右孩子编号必为 2i+1;其双亲的编号为 i/2;

（4） 二叉树的顺序存储

实现： 按满二叉树的结点层次编号，依次存放二叉树中的数据元素

（5）遍历二叉树

遍历 定义一一指按某条搜索路线遍访每个结点且不重复(又称周游)。

遍历 用途一一它是树结构插入、删除、修改、查找和排序运算的前提，是二叉树一切运算的基础和核心。

DLR——先序遍历,即先根再左再右

LDR——中序遍历,即先左再根再右

LRD——后序遍历,即先左再右再根

11.5 6.5、图

11.5.1 1、图的相关概念

（1）图是由一个顶点的集合 V 和一个顶点间关系的集合 E 组成:记为 G=(V,E) V :顶点的有限非空集合。
E :顶点间关系的有限集合(边集)。
存在一个结点 v，可能含有多个前驱结点和后继结点。
（2）图中顶点之间的连线若没有方向,则称这条连线为边,称该图为**无向图**。
（3）图中顶点之间的连线若有方向,则称这条连线为弧,则称该图为**有向图**。
（4）完全图稠密图稀疏图
在一个无向图中,如果任意两顶点都有一条直接边相连接,则称该图为无向完全图。
一个含有 n 个顶点的无向完全图中，n(n-1)/2 条边。
在一个有向图中,如果任意两顶点之间都有方向互为相反的两条弧相连接,则称该图为有向完全图。
在一个含有 n 个顶点的有向完全图中， n(n-1)条边。
若一个图接近完全图,称为稠密图。边数很少的图被称为稀疏图。
（5）度
a)顶点的度(degree)是指依附于某顶点 v 的边数,通常记为 TD(v)
结论:图中所有顶点的度=边数的两倍
b)出度入度
对有向图来说:
顶点的出度:以顶点 v 为始点的弧的数目,记为 OD(v)。
顶点的入度:以顶点 v 为终 点的弧的数目,记为 ID(v)。
顶点的度: TD(v)=ID(v) + OD(v).
（6）权 网络
与边有关的数据信息称为权
边上带权的图称为网图或网络
弧或边带权的图分别称为有向网或无向网
（7）路径
简单路径：序列中顶点不重复出现的路径
简单回路（简单环）：除第一个顶点与最后一个顶点之外，其他顶点不重复出现 的回路。

11.5.2 2、图的存储结构

（1）邻接矩阵（有向图，无向图，网图分别如何存储）
（2）邻接表（了解）
邻接矩阵：代码书写简单，找邻接点慢
邻接表：代码书写较复杂，找邻接点快

12 第 07 课 指针（国赛范围）

12.1 7.1、概念

（1）指针也是一个变量。和普通变量不同的是，指针变量里存储的数据是一个内存地址。
（2）指针变量的定义：类型名　*指针变量名
　　int *p;
（3）指针变量赋值
a） 先定义变量，再赋值
　　int a=3,*p;
　　p=&a;
b） 定义变量的同时，进行赋值。
　　int a=3,*p=&a;
注意：只能用同类型变量的地址进行赋值

12.2 7.2、引用与运算

12.2.1 1、指针的引用

引用指针时，首先要理解指针变量与普通变量的区别和对应关系。例如，定义一个指针变量“int *p;”和一个普通变量“int a;”，关于两者之间的各种引用方式对应关系如下：
（1）“p”等同于“&a”,表示的是内存地址
（2）“*p”等同于“a”,表示变量里存储的实际数据
（3）“*p=3;”等同于“a=3;”，表示变量的赋值方式。

12.2.2 2、指针的运算

如果定义的是局部指针变量，其地址就是随机的，直接操作会引发不可预测的错误。所以，指针变量一定要初始化后才能引用。
由于指针变量存储的是内存地址，所以也可以执行加法、减法运算，一般用来配合数组进行寻址操作。

12.3 7.3、指针与数组

在 C++中，数组名在一定意义上可以被看成指针。
“数组的指针”是指整个数组在内存中的起始地址。
“数组元素的指针”是数组中某个元素所占存储单元的地址。
例如，“ int a[10]，*p； p=a ；”
　　表示定义了一个指针 p，指向 a 数组在内存中的起始地址 a[0]。
一般可以使用“下标法”访问数组元素,如 a[5]； 也可以使用“地址法”访问数组元素， 因为数组名就代表数组在内存中的起始地址，也就是 a[0]的地址，如 a+4 就表

示 a[4]的地址； 当然，也可以通过“指针法”访问数组元素，通过数组的指针或者数组元素的指针访问数组元素,能使目标程序质量更高， 占用内存更少，运行速度更快。

12.4 7.4、函数指针及扩展

程序中需要处理的数据都保存在内存空间，而程序以及函数同样也保存在内存空间。C++支持通过函数的入口地址(指针)访问函数。另一方面，有些函数在编写时要调用其他的辅助函数，但是尚未确定，在具体执行时，再为其传递辅助函数的地址。

比如排序函数 sort(a,a+n,cmp)，其中的比较函数 cmp 是根据需要传递给 sort 的，就是传递了一个函数指针。

函数指针就是指向函数的指针变量，定义格式如下:

类型名(*函数名) (参数);

例如，"int(*f) (int a，int b)； "，规范来说，此处的“函数名”应该称为“指针的变量名”。

13 第 08 课 基本算法

13.1 8.1、高精度算法

计算机最初也是最重要的应用就是数值运算。在编程进行数值运算时,有时会遇到运算的精度要求特别高,远远超过各种数据类型的精度范围;有时数据又特别大,远远超过各种数据类型的极限值。这种情况下,就需要进行“高精度运算”。

高精度运算首先要处理好数据的接收和存储问题，其次要处理好运算过程中的“进位”和“借位”问题。

【题目描述】

输入 n，输出 n!的精确值，n!=1x2x3.....xn,1<n<1000

【输入】

一个整数 n

【输出】

n! 的值

【样例输入】

10

【问题分析】

假设已经计算好（n-1）!，那么对于求 n!，就是用一个整数去乘以一个高精度数。只要用 n 乘以（n-1）!的每一位（从低位开始），同时不断处理进位。

```
#include<iostream>
using namespace std;
int main( ) {
    int n, s, i, j, t, d = 1; //d 阶乘的位数
    int a[10000] = {};
    cin >> n;
    a[0] = 1;//第一个元素是 1，表示 1 的阶乘
```

```
    for (i = 2; i <= n; i++) {
        s = 0;
        for (j = 0; j < d; j++) {
            t = a[j] * i + s; //当前位置的数=之前在这个位置上的数乘以 i，然后加
上前一位数的进位
            a[j] = t % 10;    //该位置的数 即为当前值对 10 的取余
            s = t / 10;    //进位即为该位置的数 除以 10
        }
    while (s!=0) {    //如果还不是 0 说明还要向上一位进位
            a[d++] = s % 10;
            s = s / 10;
        }
    }
    for (i = d-1; i >= 0; i--)  //d-1 去掉前导零
        cout<<a[i];
    return 0;
}
```

13.2 8.2、递推算法

“递推”是计算机解题的一种常用方法。

利用“递推法”解题首先要分析归纳出“递推关系”。

比如经典的斐波那契问题，用 f(i)表示第 i 项的值，则 f(1)=0,f(2)=1，在 n>2 时，存在递推关系:f(n)=f(n-1)+f(n-2) 。

在递推问题模型中，每个数据项都与它前面的若干数据项(或后面的若干数据项）存在一定的关联，这种关联一般是通过一个“递推关系式”来描述的。求解问题时，需要从初始的一个或若干个数据项出发，通过递推关系式逐步推进， 从而推导出最终结果。

这种求解问题的方法叫“递推法”。其中，初始的若干数据项称为“**递推边界**”。

解决递推问题有三个重点:

一是建立正确的递推关系式;

二是分析递推关系式的性质;

三是根据递推关系式编程求解。

13.3 8.3、分治算法

“分治”是一种常用的解题策略。它是将一个难以直接解决的大问题 ,分解成若干规模较小的、相互独立的、相同或类似的子问题，分而治之，再合成得到问题的解。根据“平衡子问题”的思想，一般会把问题分解成两个规模相等的子问题， 也就是“**二分法**”，比如经典的二分查找(由半查找)问题。

例：找伪币。

【问题描述】

给出 16 个一模一样的硬币,其中有 1 个是伪造的，并且那个伪造的硬币比真的硬币要轻一些，本题的任务是找出这个伪造的硬币。为了完成这一任务，将提供一台可用来比较两组硬币重量的仪器，可以知道两组硬币孰轻孰重。

【问题分析】

方法 1 **穷举法**
依次比较硬币 1 与硬币 2、硬币 3 和硬币 4、硬币 5 和硬币 6……最多通过 8 次 比较来判断伪造币的存在并找出这个伪币。
方法 2 **二分法**
把 16 个硬币的情况看成一个大问题。
第一步,把这一大问题分成两个小问题，随机选择 8 个硬币作为第一组(A 组),剩下的 8 个硬币作为第二组(B 组)。
第二步,利用仪器判断伪币在 A 组还是在 B 组中,如果在 A 组中则再把 A 组中的 8 个硬币 随机分成 2 组,每组 4 个再去判断...这样,只要(必须)4 次比较一定能找出伪币。
方法 3 **三分法**
把 16 个硬币分成 3 组(A 组 5 个、B 组 5 个、C 组 6 个),利用仪器比较 A、B 两组， 一次就可以判断出伪币在 A、B 、C 哪一组中。假如在 C 组中,则再把 C 组 中的 6 个分成 3 组(2 个、2 个、2 个)，再用仪器比较一次判断出在哪一组。然后再比较 1 次就能判断出 2 个硬币中哪个是伪币。这样，只要 2~3 次比较便能找出伪币。

13.4 8.4、贪心算法（国赛范围）

贪心法的基本思想
贪心法是从问题的某个初始解出发，采用逐步构造最优解的方法，向给定的目标前进。在每一个局部阶段，都做一个“看上去最优的决策，并期望通过每一次所做的局部最优选择产生出一个全局最优解。做出贪心决策的依据称为“贪心策略”。要注意的是，贪心策略一旦做出，就不可再更改。与递推不同的是，贪心严格意义上说只是一种策略或方法，而不是算法。推进的每一步不是依据某一个固定的递推式，而是做一个当时“看似最佳”的贪心选择(操作),不断将问题归纳为更小的相似子问题。所以，归纳、分析、选择正确合适的贪心策略，是解决贪心问题的关键。

13.5 8.5、搜索算法（宽度优先搜索、深度优先搜索）

13.5.1 1、宽度优先搜索

宽度优先搜索的基本思想
宽度优先搜索（Breadth First Search, BFS），简称宽搜，又称为广度优先搜索。它是从初始结点开始，应用产生式规则和控制策略生成第一层结点，同时检查目标结点是否在这些生成的结点中。若没有，再用产生式规则将所有第一层结 点逐一拓展，得到第二层结点，并逐一检查第二层结点是否包含目标结点。若没有， 再用产生式规则拓展第二层结点。如此依次拓展，检查下去，直至发现目标
结点为止。如果拓展完所有结点，都没有发现目标结点，则问题无解。
在搜索的过程中，宽度优先搜索对于结点是沿着深度的断层拓展的。如果要 拓展第 n+1 层结点，必须先全部拓展完第 n 层结点。对于同层结点来说，它们对于问题的解的价值是相同的。所以，这种搜索方法一定能保证找到最短的解序列。也就是说，第一个找到的目标结点，一定是应用产生式规则最少的。因此，宽度优先搜索算法适合求最少步骤或者最短解序列这类最优解问题。

13.5.2 2、深度优先搜索

深度优先搜索(Depth First Search，DFS),简称**深搜**，其状态“退回一步”的顺序符合“后进先出”的特点，所以采用“栈”存储状态。深搜的空间复杂度较小，因为它只存储了从初始状态到当前状态的条搜索路径。但是深搜找到的第一个解 不一定是最优解，要找最优解必须搜索整棵“搜索树”。所以，深搜适用于要求所有解方案的题目。
深搜可以采用直接递归的方法实现，其算法框架如下:

```
void dfs (dep: integer，参数表);
{
    自定义参数;
    if (当前是目标状态) {
        输出解或者作计数、评价处理;
    } else
        for (i = 1; i <= 状态的拓展可能数; i++)
            if (第 i 种状态拓展可行) {
                维护自定义参数;
                dfs(dep + 1, 参数表);
            }
}
```

13.6 8.6、动态规划算法

动态规划是一种将问题实例分解为更小的、相似的子问题,并存储子问题的解 而避免计算重复的子问题,以解决最优化问题的算法策略。动态规划实际上就是 一种排除重复计算的算法,更具体的说,就是用空间换取时间。
能采用动态规划求解的问题的一般要具有 3 个性质:
(1) 最优化原理:问题的最优解所包含的子问题的解也是最优的
(2) 无后效性:即某阶段状态一旦确定,就不受这个状态以后决策的影响。
(3)有重叠子问题:即子问题之间是不独立的,一个子问题在下一阶段决策中可能被多次使用到。

动态规划问题的一般解题步骤：

1、判断问题是否具有最优子结构性质,若不具备则不能用动态规划。
2、把问题分成若干个子问题(分阶段)。
3、建立状态转移方程(递推公式)。
4、找出边界条件。
5、将已知边界值带入方程。