

# 码蜂C++程序设计--基础篇

## 第一章 初识C++

### 1. 软件的安装及配置

1. 软件安装
2. 语言选择
3. 字体大小

### 2. 第一个程序及注意事项

1. 部分关键单词

单词	含义
include	包含、包括
input	输入
output	输出
io	输入输出
stream	河流, 小河
use	使用
space	空间, 空白
int	整数 (integer)
main	主要的
end	结束
line	线条, 行

2. "hello, world!"

```
#include<iostream> //输入输出流工具包

using namespace std; //防止命名冲突

int main(){//主函数, 程序的入口, 花括号不能丢

    cout<<"hello,world!";//c++的out(输出)功能, 类似龟派气功对外输出

    return 0; //后续学习到函数再讲解

} //花括号成对出现, 不能丢
```

3. 开学小黄要买文具，铅笔一盒20支，每支铅笔1元，买了6盒，一个文具盒8元，买了6个。问：一共要付多少钱？

**注：**数学方法如何计算？编程呢？

4. 程序的保存、编译及运行&编译并运行
5. 几个需要补充的小问题

1. 缩进 (Tab) 键可以使代码更加美观，可读性更强

2. insert键问题
3. 程序报错信息阅读 (一般要看当前行与上一行)
4. 点击行号出现红色 (断点)
5. Ctrl+滚轮, Ctrl+C、Ctrl+V、Ctrl+Z、Ctrl+Y
6. 课堂练习

实验一下数学中的加减乘除在代码中的运行结果是否一致?

注: + - \* /

### 3.C++的基本运算

1. C++中常见的运算

运算符	描述	案例
+	两数相加	5+2
-	两数相减	5-2
*	两数相乘	5*2
/	两数相除	5/2
%	求整除后的余数	5%2

2. 常见运算案例

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

int main(){
    cout<<5+2<<endl;
    cout<<5-2<<endl;
    cout<<5*2<<endl;
    cout<<5/2<<endl;
    cout<<5%2<<endl;
}
```

#### 注意:

1. C++中两个整数运算结果一定是整数
2. 整数和小数运算或小数与小数运算一定得到小数 (精度) ,

eg:

5/2=2,3/5=0

$5/2.0=2.5, 5.0/2=2.5$

3. %表示整除后剩下的余数, eg:

$5\%2=1, 2\%5=2;$

#### 4. 练习

1) 求长方形的周长与面积

**描述:**

假设一个长方形的两条边分别为2和8, 请编程计算出该长方形的周长跟面积

第一行输出周长, 第二行输出面积

**提示:**

周长 = (长+宽)  $\times 2$

面积 = 长 $\times$ 宽

2) 问题: 王师傅应付多少钱?

**描述:**

张阿姨去集市卖鸡, 公鸡50元/只, 母鸡60元/只。王师傅买了22只公鸡和15只母鸡, 请问王师傅一共付了多少钱? 平均一只鸡多少钱? 请编程求解:

**输出:**

第一行输出总价, 第二行输出一只鸡平均价格

**提示:**

平均值=总价/总鸡数

3) 问题: 期末考试成绩统计:

描述: 期末考试到了, 小黄语文、数学、英语分数分别是: 100、95、90.请编程计算出三门课的总分、和平均分显示在两行内 (注意会有小数产生, 要保留小数)

## 第二章 变量

### 1. 什么是变量

在数学中, 我们用字母来代表数字, 这就是数学中的“变量”概念。

1. 一只青蛙四条腿, 两只青蛙 \*4条腿, 三只青蛙 \*4条腿, 四只青蛙 \*4条腿, n只青蛙  
\*4条腿... ...

2. 小黄早饭买了n个包子, 每个包子1.5元, 小黄早饭吃了多少钱?

3. 小猴子发现s个香蕉, 但是它每次只能拿m根, 请说出下面的表格内应该填多少:

搬香蕉次数	1	2	3	4	n
剩余香蕉个数					

4. 长方形的长是x, 宽是y, 周长c是多少? 面积s呢?

### 2.C++中的变量

与数学中的变量类似, 都是用一个字母或单词来代表一个数据。

1. 变量的命名规则:

1、变量名只能是字母 (A-Z,a-z) 、数字(0-9)或下划线。

2、第一个字母不能是数字。例如: 2Lerver这不是一个合法的C++变量。

3、不能是C++关键字。例如: 不能用int这个单词来命名一个变量。

4、区分大小写。例如：int a 和 int A是两个不同的变量。

5、在同一个作用域，不能存在相同的变量名。例如：int a 和 int a不能在同一个作用域存在。

**注：**解释说明作用域的概念

2. 与数学中不同的是，C++中的变量必须要先声明（定义）才可以使用

1. 解释：定义指告诉计算机，我们需要使用一个什么样的名字来存储某个**类型的**数据，如 int n，表示我们将用字母n来存储一个int类型的数据

2. 案例：

小黄早饭买了x个包子，每个包子y元，已知x=20,y=2，请用编程来计算小黄早饭吃了多少钱？

```
#include<iostream>

using namespace std;

int main(){
    int x=20;
    int y=2;
    cout<<x*y<<endl;
}
```

3. 在C++中，如果变量定义了但并没有赋值，则变量的值是无法确定的，这是因为所有的数据都要依附在内存中，而在定义了一个变量的同时，计算机会分配一个内存空间（根据数据类型）给对应的变量，若没有进行赋值操作，则该空间上原有的数据将会被保存下来，所以是不确定的。

**注意：**要区分理解**什么是定义，什么是赋值，变量能否重复定义和变量能否重复赋值问题**

4. 变量的不同类型

1. 整型变量：short、int、long、long long

2. 浮点型变量：float、double

3. 字符型变量：char

4. 字符串变量：char\*、string等。

5. 计算机中的单位：

1 byte= 8 bit

1 KB= 1024 byte

1 MB= 1024 KB

1 GB= 1024 MB

1 GB= 1024 \* 1024 \* 1024 \* 8 = 85.9亿 bit

6. 不同数据类型的取值问题，以int为例：

定义有符号的整型变量，占32个bit。能表示的范围是： -2147483648 ~ 2147483647

**注：**一个bit可以表示0或1，所以int占位32最大可以表示  $2^{32}$  这么多种情况，转化为10进制即为 2147483648\*2，此处只有一半是因为下文所述原因

1111 1111 1111 1111 1111 1111 1111 1111

从左往右最左边的第一个数表示符号，0表示正数，1表示负数。

**注意：题目出现10位数的时候数据类型要设为int**

7. 其它数据类型的取值：

类型	定义	大小	范围
char	字符型	1 byte	-128~127
short(int)	短整型	2 byte	-32768~32767 ( $-2^{15}$ ~ $2^{15}-1$ )
int	整形	4 byte	-2147483648 ~ 2147483647 ( $-2^{31}$ ~ $2^{31}-1$ )
long	长整形	4 byte(32bit 系统) 或 8 byte(64bit 系统)	参考 int 和 long long 的范围
long long	长整形	8 byte	-9223372036854775808 ~ 9223372036854775807 ( $-2^{63}$ ~ $2^{63}-1$ )
float	实型	4 byte	- (10 的 38 次方) ~ 10 的 38 次方
double	双精度	8 byte	- (10 的 308 次方) ~ 10 的 308 次方

8. 浮点数（小数）问题：

**float** 单精度浮点型，占用32bit，最多能表示8个有效位。范围是  
1.175494351\*10<sup>-38</sup>—3.402823466\*10<sup>38</sup>

**double** 双精度浮点型，占用64bit，最多16个有效位。范围是：  
2.2250738585072014\*10<sup>-308</sup>—1.7976931348623158\*10<sup>308</sup>

9. 课堂练习

- 一个长方形的两条边分别是为x=2.5、y=3，请通过编程来实现计算该长方形的面积和周长。要求：用S代表面积，C代表周长
- 小黄向老板请了n天的假 (n=10),请编程计算小黄请了多少个小时的假，请了多少分钟？要求：用h表示小时，用m表示分钟
- 一个正方形的周长为18，边长是多少？面积呢？要求：用l表示边长，用s表示面积。

10. 课后作业：

- 小黄挖了20个萝卜，小狗挖的数量是小黄的3倍，小兔挖到的数量比小狗少7个。问：小狗挖了多少？小兔呢？平均每人挖了多少？要求：用x表示小狗的萝卜数量，y表示小兔的萝卜数量，a表示平均每人多少个的数量。
- 分跳绳  
描述：学校新买了m根跳绳 (m=186),每个班分n根 (n=55)，最多可以分给几个班级？还剩多少根？  
要求：使用x表示可以分给几个班，y表示剩余的跳绳数量。

# 第三章 键盘输入及格式化输入输出

## 1.为什么要从键盘读取数值？

以计算边长和面积的题目为例：一个正方形的周长为18，边长是多少？面积呢？要求：用l表示边长，用s表示面积。

```
#include<iostream>

using namespace std;

int main(){
    double c = 18;
    double l = c/4.0;      //double类型可防止无法整除时精度丢失
    double s = l*l;
    cout<<l<<" "<<cout<<s<<endl;
    return 0;
}
```

**存在问题：**如果数据发生变化，则需要修改源代码中周长的数值，失去灵活性。

**解决方式：**从键盘读入长方形的周长，这样就可以根据输入的数值进行计算了。

**代码：** cin>>c;

**修改后代码：**

```
#include<iostream>

using namespace std;

int main(){
    double c ;
    cin>>c;
    double l = c/4.0;      //double类型可防止无法整除时精度丢失
    double s = l*l;
    cout<<l<<" "<<s<<endl;
    return 0;
}
```

```
}
```

**注：**需要多个输入时可以通过多个cin或一个cin加多个“>>”实现。如

```
int x ;  
  
int y ;  
  
cin >> x >> y;
```

**注：2** 输入时只要在两个数值之间+空格即可。但是需要注意，如果变量类型是char（字符）时，计算机会将空格认为是一个字符

## 2.课堂案例

**解题的关键步骤：**

1. 读懂题意
2. 明确需要几个变量需要输入，分别是什么类型
3. 明确需要几个变量需要输出，分别是什么类型
4. 建立数学模型，思考解题步骤
5. 通过编程实现

### 6. 求长方形的周长和面积：

**描述：**从键盘读入2个整数，分别代表长方形的长和宽，请计算长方形的周长和面积。

**输入：**从键盘上读入2个整数

**输出：**输出两行，第一行表示周长，第二行表示面积

### 7. 分跳绳

**描述：**学校新买了m根跳绳，每个班分n根（m>=n），最多可以分给几个班级？还剩多少根？

**输入：**两个整数，用空格隔开，分别代表总共采购的数量和分配到各班级的数量

**输出：**两个整数，用空格隔开，分别表示可以分给几个班和剩余的跳绳数量。

## 3.课后作业

### 1.题目描述

时钟上面的时针从m时走到n时旋转了多少度？（m<=n，且m和n都是1~12之间的整数）



**输入：** 2个整数m和n

**输出：** 一个整数代表时针旋转的度数

**样例输入：** 1 4

**样例输出：** 90

**提示：** 时针一小时转多少度？

## 2. 题目描述

文具店的水笔的单价是一个整数，小黄从文具店买了 $x$ 支水笔花了 $y$ 元。请问如果你要从该文具店购买5支水笔需要付多少钱？

**输入：** 两个整数 $x$ 和 $y$ ，用空格隔开，分别代表小黄买的水笔的支数和他付给老板的金额

**输出：** 一个整数，代表如果你买5支水笔，应该付给老板的金额

**样例输入：** 10 20

**样例输出：** 10

**提示：** 一支水笔多少钱？

## 4. 四舍五入保留小数的方法

1. 多引入一个头文件，#include<iomanip>, 用于支持保留小数的函数；

2. 输出格式：

```
cout<<fixed<<setprecision(2)<<d<<endl;
```

### 3. 关键单词

单词	意义
fixed	固定的
set	设置
precision	精度

## 5.课堂练习

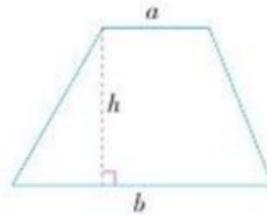
### 1. 求梯形面积

#### 题目描述

梯形面积的求解公式为  $S = (a + b) * h / 2$ 。从键盘读入一个梯形的上底a、下底b和高h，请计算梯形的面积。（结果保留1位小数）

如果用  $S$  表示梯形的面积，用  $a$ 、 $b$  和  $h$  分别表示梯形的上底、下底和高，上面的公式可以写成：

$$S = (a + b) \times h \div 2$$



#### 输入

三个整数a、b、h

#### 输出

梯形的面积

#### 样例输入

```
2 3 5
```

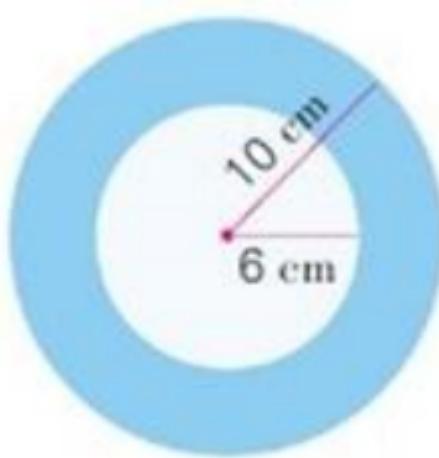
#### 样例输出

```
12.5
```

### 2. 求圆环面积

#### 题目描述

如下图所示的圆环铁片，中间是空心的，已知圆环外圆的半径是r1厘米（如：10cm），内圆半径是r2厘米（如：6cm），请编程计算该铁片的面积。（外圆面积 - 内圆面积，假设 $\pi=3.14$ ，结果保留2位小数）



#### 输入

2个整数r1、r2，分别代表外圆和内圆的半径。

#### 输出

铁片的面积。

### 样例输入

10 6

### 样例输出

200.96

## 3. 求花坛面积

### 题目描述

龙湖校区有一个圆形花坛，量得花坛周围的篱笆长是x米，请问该花坛的面积是多少平方米？（假设 $\pi=3.14$ ）（5.2.99）

### 输入

一个小数x

### 输出

花坛的面积（结果保留2位小数）

### 样例输入

18.84

### 样例输出

28.26

### 提示

圆的面积和周长求解公式分别如下：

圆的面积 $S = \pi * \text{半径} * \text{半径}$ ；（π读作pài，π = 3.1415926...，在具体的题目中π的值精确到小数点后多少位，取决于具体的题目）

圆的周长 $C = \pi * 2 * \text{半径}$ ；

## 第四章 编程中的计算（以拆数字为例）

### 1. 拆位

**拆位：**拆出一个整数n的每一个数位上的数字

**描述：**从键盘读入一个两位数，请问两位数的个位和十位的和是多少？

**输入：**一个两位整数

**输出：**该数个位和十位的和

**样例输入：**23

**样例输出：**5

**拆位原理：理解一个整数的构成原理：**

1. 两位数n的组成：十位 $*10 +$ 个位 $*1$

例如： $23=2*10+3*1$

2. 三位整数n的组成：百位\*100 + 十位\*10 + 个位\*1

例如： $523=5*100 + 2*10 + 3$

**拆位示范：**拆一个两位数的十位和个位并求和

**思路：**

int n;

n的十位= $n/10$ ,因为整数与整数计算得到整数，即只保留了十位上的数字

n的个位= $n \% 10$ ，因为一个两位数对10取余只能得到不够10的数字，即个位

**示例代码：**

```
#include<iostream>
using namespace std;
int main(){
    int n, g , s ;//n为原始数字, g为存储个位, s为存储十位
    cin >> n ;//获取n的值
    s= n / 10 ;
    g= n % 10 ;
    cout << s + g << endl ;
    return 0 ;
}
```

## 2. 拆位的规律

**重点：**归纳法找出“/”和“%”的区别

例子：拆出一个三位数的各个数位

$233 / 10 =$

$233 / 100 =$

$233 / 1000 =$

**结论：**

当我们使用/运算时，相当于\*\*去掉\*\*原数字的尾数

1. /10, 去掉1个尾数

2. /100, 去掉2个尾数

3. /1000, 去掉3个尾数

$233 \% 10 =$

$233 \% 100 =$

$233 \% 1000 =$

**结论：**

当我们使用 % 运算时，相当于**获取**原数字的尾数

1. %10, 获取1个尾数

2. %100, 获取2个尾数

### 3. %1000, 获取3个尾数

**推论：**如果想拆出一个三位数（以233为例）的各个位，可以

1. 百位 =  $n$  去掉2个尾数即 =  $n / 100$ ；
2. 个位 =  $n$  得到1个尾数即 =  $n \% 10$ ；
3. 十位 = 去1个尾数之后再得一个尾数（先得到23，再取3）  
 $= n / 10 \% 10$
4. 十位或者等于得到2个尾数后去掉一个尾数（先得到33，再取3）  
 $= n / 100 / 10;$

## 3.课堂练习

### 1. 求任意三位数各个数位上的数字之和

#### 题目描述

对于一个任意的三位自然数X，编程计算其各个数位上的数字之和S。

#### 输入

输入一行，只有一个整数x( $100 \leq x \leq 999$ )

#### 输出

输出只有一行，包括1个整数

#### 样例输入

```
123
```

#### 样例输出

```
6
```

### 2. 输入一个三位数，把个位和百位对调后输出

#### 题目描述

输入一个三位自然数，然后把这个数的百位数与个位数对调，输出对调后的数

#### 输入

输入一行，只有一个整数x( $100 \leq x \leq 999$ )。

#### 输出

输出只有一行，包括1个整数。

#### 样例输入

```
123
```

#### 样例输出

```
321
```

### 3. 加密四位数

### 题目描述

某军事单位用4位整数来传递信息，传递之前要求先对这个4位数进行加密。加密的方式是每一位都先加上5然后对10取余数，再将得到的新数颠倒过来。

例如：原数是1379，那么每位加5对10取余数的结果为6824，然后颠倒该数，得到新数：4286。

再比如：原数是2570，那么每位加5对10取余数的结果为7025，然后颠倒高数，得到新数：5207。

请根据加密要求，写出加密算法！

### 输入

加密前的4位数

### 输出

加密后的结果

### 样例输入

```
1379
```

### 样例输出

```
4286
```

## 4.课后作业

### 1. 算算和是多少

#### 题目描述

输入一个三位正整数，然后与它倒过来的数相加，输出和。

如：输入167，则和为 $167+761=928$

#### 输入

只有一行，一个三位正整数。

#### 输出

一个正整数

#### 样例输入

```
167
```

#### 样例输出

```
928
```

### 2. 倒序输出一个四位整数

#### 题目描述

任意读入一个四位整数，颠倒后输出。

#### 输入

输入一行，只有一个整数 $x(1000 \leq x \leq 9999)$ 。

#### 输出

输出只有一行，包括1个整数。

#### 样例输入

```
4567
```

#### 样例输出

```
7654
```

### 3. 求一个5位数的各个位之和

#### 题目描述

从键盘读入一个5位的正整数，请求出这个5位数的各个位之和；

#### 输入

一个5位的正整数n

#### 输出

这个5位数的各个位之和

#### 样例输入

```
12345
```

#### 样例输出

```
15
```

## 第五章 分支

### 1. 什么是分支？

通过以下案例来理解什么是分支以及分支的意义所在。

#### 1. 冷饮的价格

#### 题目描述

小黄去冷饮店买冰激凌，如果买10个以上或者10个，2元/个，10个以下，2.2元/个，请从键盘读入小黄的购买数量，计算小黄应付的价格！

#### 输入

一个整数，代表小黄购买的冰激凌的数量 ( $n \leq 100$ )

#### 输出

小黄应付的金额，金额保留1位小数！

#### 样例输入

```
20
```

#### 样例输出

```
40.0
```



双分支问题，一个问题有两种不同的情况

## 2. 冷饮的价格2

### 题目描述

小黄夏天去买冰棍，老板说买30个及以上1元/个，20~29个1.2元/个，10~19个1.5元/个，10个以下1.8元/个！请从键盘读入小黄买冰棍的数量，计算小黄应该付的价格（价格保留1位小数）！

### 输入

一个整数n代表小黄购买的冰棍的数量

### 输出

小黄应付的金额

### 样例输入

30

### 样例输出

30.0



多分支问题：一个问题有大于2中不同的情况

## 2. 双分支的基本语法

### 1. 基本语法结构：

```

if(判断条件){
    /*条件成立执行
    这里的内容*/
} else{
    /*条件不成立
    执行这里的内容*/
}

```

1. if: 如果, 假如
  2. else: 否则
  3. true: 真
  4. false: 假
2. 什么是判断条件

判断条件是指一个**条件表达式**, 通常是判断两者之间的关系:

如果为真 (true) 则表达式成立

如果为假 (false) 则表达式不成立

如 int x = 10 ; 请判断下面条件表达式的结果:

关系判断	举例	含义
>	x>10	是否大于
>=	x>=10	是否大于或等于
==	x==10	是否等于 (=为赋值)
!=	x!=10	是否不相等
<	x<10	是否小于
<=	x<=10	是否小于或等于

### 3.课堂案例

#### 1. 行李托运价格

##### 题目描述

某车站行李托运收费标准是: 10公斤或10公斤以下, 收费2.5元, 超过10公斤的行李, 按每超过1公斤增加1.5元进行收费。试编一程序, 输入行李的重量, 算出托运费。

##### 输入

输入只有一行, 包括1个整数。

##### 输出

输出只有一行, 包括1个数。 (保留两位小数)

##### 样例输入

10

##### 样例输出

## 2. 心系南方灾区

### 题目描述

2008年年初我国南方正在承受百年不遇的大雪、冻雨灾害。北京市已经开始了面向全体市民的捐款捐物活动，并组织运力，以最快速度将这些救灾物资运送到灾区人民的手中。已知救灾物资中有 $m$ 件大衣 ( $10000 \leq m \leq 2000000$ )，一辆卡车一次最多可以运走 $n$ 件 ( $2000 \leq n \leq 10000$ )。请你编写程序计算一下，要将所有的大衣运走，北京市政府最少需要调动多少辆卡车参与运送。

### 输入

只有两个整数  $m$ 、 $n$ 。

### 输出

只有一个整数，表示需要的卡车数量。

### 样例输入

```
10000 2000
```

### 样例输出

```
5
```

## 4. 课后作业

### 1. 恐龙园买门票

#### 题目描述

恐龙园买门票，身高低于1.3米购儿童票(60元)，否则成人票120元。试编写一个程序，输入身高，输出相应的门票价格。

#### 输入

一行，一个人的身高。

#### 输出

一行，一个整数

#### 样例输入

```
1.1
```

#### 样例输出

```
60
```

### 2. 判断能否构成三角形

#### 题目描述

输入三个整数，表示3条线段的长度，判断这三条线段能否构成三角形。能构成就输出'Yes'，否则输出'No'。

三角形的判断标准是：任意两边之和要大于第三边，比如有一个三角形的三条边分别为：3 5 7，这个三角形的三条边就满足 $3 + 5 > 7$ 且 $3 + 7 > 5$ ，且 $5 + 7 > 3$ ，因此这三条边能够构成三角形；

再比如，一个三角形的三条边为3 8 5，那么因为 $3+5$ 不满足大于8，就不能构成三角形。

### 输入

三个整数

### 输出

Yes or No

### 样例输入

```
3 8 5
```

### 样例输出

```
No
```

### 提示

构成三角形要求：两边之和大于第三边

## 3. 扩建鱼塘

### 题目描述

有一个尺寸为 $m * n$ 的矩形鱼塘，请问如果要把该鱼塘扩建为正方形，那么它的面积至少增加了多少平方米？

### 输入

两个整数 $m$ 和 $n$

### 输出

一个整数，代表鱼塘面积增加的值

### 样例输入

```
5 3
```

### 样例输出

```
10
```

## 4. 最多能倒多少杯水？

### 题目描述

小黄所在的学校引入了电水箱为同学们烧开水。已知电水箱的容量为 $n$ 升 ( $n \leq 10L$ )，同学们带的杯子平均容量为 $x$ 毫升 ( $x$ 在100~300之间)，请问烧一箱开水，最多能倒多少杯 (不足1杯算1杯)。

### 输入

2个整数 $n$ 和 $x$ ； $n$ 代表电水箱的总容量 (单位为升L)， $x$ 代表平均1个杯子的容量 (单位为毫升ml)

### 输出

一个整数，代表最多能够倒多少杯水 (不足1杯算1杯)

### 样例输入

```
1 120
```

### 样例输出

9

#### 5. 两数比大小

##### 题目描述

有A, B两个不相等的数, 请将其中较大数打印出来。

##### 输入

输入只有一行, 包括2个整数。之间用一个空格分开。输出只有一行 (这意味着末尾有一个回车符号), 包括1个整数。

##### 输出

输出只有一行 (这意味着末尾有一个回车符号), 包括1个整数。

##### 样例输入

45 78

##### 样例输出

78

## 第六章 多分支

### 1. 多分支判断的语法

```
if(判断条件){  
    代码块;  
}else if(判断条件){  
    代码块;  
}else if(判断条件){  
    代码块;  
}...{  
    代码块;  
}else {  
    代码块;  
}
```

#### 注意:

1. else if 后面必须加判断条件, else 后面不可以加
2. else if后面的 else 并非必要加
3. 多分支中至多只能有一个条件成立, 即**一旦有一个条件成立, 其余的条件均不再判断**, 这里区别于多个 if 平级判断

#### 回顾: 双分支语法

```
if(判断条件){  
    条件成立执行这里;  
}else{  
    条件不成立执行这里;  
}
```

## 2.课堂案例

### 1. 冷饮的价格2

#### 题目描述

小黄夏天去买冰棍，老板说买30个及以上1元/个，20~29个1.2元/个，10~19个1.5元/个，10个以下1.8元/个！请从键盘读入小黄买冰棍的数量，计算小黄应该付的价格（价格保留1位小数）！

#### 输入

一个整数n代表小黄购买的冰棍的数量

#### 输出

小黄应付的金额

#### 样例输入

```
30
```

#### 样例输出

```
30.0
```



### 2. 判断成绩等级

#### 题目描述

输入某学生成绩，如果86分以上(包括86分) 则输出“VERY GOOD”，如果在60到85之间的则输出“GOOD”(包括60和85)，小于60的则输出“BAD”。

#### 输入

输入只有一行，包括1个整数。

#### 输出

输出只有一行（这意味着末尾有一个回车符号）。

## 样例输入

```
80
```

## 样例输出

```
GOOD
```

### 3. 找出最经济型的包装箱型号

#### 题目描述

已知有A, B, C, D, E五种包装箱，为了不浪费材料，小于10公斤的用A型，大于等于10公斤小于20公斤的用B型，大于等于20公斤小于40公斤的用C型，大于等于40公斤的小于50公斤的用D型，大于等于50公斤小于80公斤的用E型。现在输入一货物的重量（小于80公斤），找出最经济型的包装箱型号。

#### 输入

输入只有一行，包括一个整数。

#### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个字符。

## 样例输入

```
8
```

## 样例输出

```
A
```

### 4. 求三个数的最大数

#### 题目描述

已知有三个不等的数，将其中的最大数找出来。

#### 输入

输入只有一行，包括3个整数。之间用一个空格分开。

#### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

## 样例输入

```
1 5 8
```

## 样例输出

```
8
```

### 3.课后作业

#### 1. 汉译英

##### 题目描述

输入某个整数，如果输入的整数在1-9范围内，则输出相对应的单词，否则输出'out'

##### 输入

一行，一个整数n。

##### 输出

整数n相对应的小写英文单词或'out'。

##### 样例输入

1

##### 样例输出

one

#### 2. 公交卡充值问题

##### 题目描述

小黄去公交卡充值中心为自己的公交卡充值，公交充值中心搞了一个充值优惠活动，活动详情如下：

- (1) 充值200元~299元，赠送50元余额到卡中；
- (2) 充值300元~499元，赠送100元余额到卡中；
- (3) 充值500元及500元以上，赠送200元余额到卡中；
- (4) 充值200元以下，则没有赠送活动；

比如：小黄如果充值350元，那么实际卡中到账的金额将会是450元（350元充值 + 100元赠送）。

请编程帮助公交卡充值中心，根据客户的充值金额，计算实际应当到账的金额？

##### 输入

一个整数n，代表小黄的充值金额（n是1~999之间的整数）

##### 输出

一个整数，代表实际到账的金额

##### 样例输入

200

##### 样例输出

250

#### 3. 求数的量级

##### 题目描述

有一个很大的整数n ( $n \geq 10000$ 且 $n \leq 9999999999$ )，请问该数的最高位是什么量级的，输出该量级的拼音？

可选单位：万 (wan) 、十万 (shi wan) 、百万 (bai wan) 、千万 (qian wan) 、亿 (yi) 、十亿 (shi yi) 。

如：n=123456789，则输出：yi

(4.2.15)

### 输入

一个很大的整数n (n>=10000且n<=9999999999)

### 输出

n量级的拼音

### 样例输入

```
123456789
```

### 样例输出

```
yi
```

## 4. 求四个数字中的最大数

### 题目描述

已知有四个不等的数，将其中的最大数找出来。

### 输入

输入只有一行，包括4个整数。之间用一个空格分开。

### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

### 样例输入

```
1 9 8 6
```

### 样例输出

```
9
```

## 5. 小黄暑假的零花钱

### 题目描述

小黄同学的妈妈在期末考试之后决定根据小黄的考试成绩奖励小黄不同的暑假零花钱，如果考试成绩在90分以上（包括90分），零花钱是成绩的3倍，如果考试成绩在80~90之间（包括80不包括90），零花钱是成绩的2倍，如果成绩在70~80之间（包括70不包括80），零花钱就是成绩的分数值，如果成绩在70以下，那么暑假只有50元的零花钱。

请从键盘读入小黄同学的考试成绩（0~100之间的整数），根据考试成绩计算小黄暑假应得的零花钱。

### 输入

键盘读入一个整数n代表小黄同学的考试成绩（0~100之间）

### 输出

小黄暑假的零花钱的金额

## 样例输入

90

## 样例输出

270

# 第七章 多分支、双分支和多if的应用

## 1.三者的区别

差异	双分支	多分支	多个if
语法	if(判断条件){ /*条件成立执行这里的內容*/ }else{ /*条件不成立 执行这里的內容*/ }	if(判断条件){ 代码块; }else if(判断条件){ 代码块; }else if(判断条件){ 代码块; }...{ 代码块; }else { 代码块; }	if(判断条件){ 执行语句; } if(判断条件){ 执行语句; } if(判断条件){ 执行语句; }
含义	问题分为两种不同时成立的情况，即if成立执行if，否则执行else	问题分为多种情况，但最多只有一个成立	问题分为多种情况，可以多种情况同时成立

### a.课堂练习

#### 1. 判断是否适合晨练

##### 题目描述

夏天到了，气温太高，小黄的爷爷每天有晨练的习惯，但有时候温度不适合晨练；小黄想编写一个程序，帮助爷爷判断温度是否适合晨练，输入温度t的值，判断其是否适合晨练，适合晨练输出OK，不适合输出NO。（20 <= t <= 30，则适合晨练OK，否则不适合NO）

##### 输入

一个整数代表当天的温度

##### 输出

OK或者NO

##### 样例输入

22

##### 样例输出

OK

#### 2. 判断奇偶数

##### 题目描述

输入一个整数，判断是否为偶数。是输出"yes"，否则输出"no"。

##### 输入

**输入**只有一行，包括1个整数。

### **输出**

输出只有一行。（注意输出格式，具体请看下方提示）

### **样例输入**

```
2
```

### **样例输出**

```
y e s
```

### **提示**

要注意空格！！！！！！！

## 3. 判断一个整数能被2、3、5、7中的谁整除

### **题目描述**

从键盘读入一个整数n，请问n能够被2、3、5、7中哪些数整除，从小到大依次输出n能够整除的数，每行一个。

如：

输入：20

输出：

```
2
```

```
5
```

### **输入**

一个整数n (n<=10000)

### **输出**

n能够整除的数

### **样例输入**

```
20
```

### **样例输出**

```
2
```

```
5
```

## b.小问题

1. 能用多分支解决的问题能不能用多个if解决？
2. 能用多个if解决的问题能否使用多分支解决？为什么？

## 2.课堂案例

### **解题关键步骤：**

1. 判断问题应该使用双分支、多分支还是多if，以及为何这样选择
2. 思考问题分为几种情况，每种情况如何判断
3. 考虑每种情况该如何计算

## 4. 编程实现

### 练习

#### 1. 要买几个止咳糖浆？

##### 题目描述

小黄生病了，妈妈去给小黄买儿童止咳糖浆。一瓶儿童止咳糖浆的规格及用法如下所示，一般小黄咳嗽需要5天才能痊愈。止咳糖浆规格：每瓶120毫升，每日3次，10岁以上儿童：每次25毫升，7-10岁儿童：每次15毫升，3-6岁儿童：每次5毫升。请根据止咳糖浆的规格以及小黄的年龄计算，写一个程序计算如果小黄要痊愈，妈妈至少要买几瓶止咳糖浆？

##### 输入

一个整数，代表小黄的年龄（小黄的年龄在3岁以上）

##### 输出

一个整数，代表妈妈最少需要购买的糖浆瓶数

##### 样例输入

```
5
```

##### 样例输出

```
1
```

#### 2. 求三个数的最大数

##### 题目描述

已知有三个不等的数，将其中的最大数找出来。

##### 输入

输入只有一行，包括3个整数。之间用一个空格分开。

##### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

##### 样例输入

```
1 5 8
```

##### 样例输出

```
8
```

讲解“打擂台”找最大数的方法

## 3. 用分支对数字进行排序

#### 1. 两个数字的排序

例题：请将两个数按照由小到大排序后输出

##### 题目描述

从键盘读入两个整数a和b，将这两个数按照由小到大的顺序输出；

### 输入

输入任意的两个整数，用空格隔开

### 输出

输出只有1行，按照由小到大的顺序输出2个整数，用空格隔开

### 样例输入

```
2 1
```

### 样例输出

```
1 2
```

### 思路1：

不真正排序，双分支穷举所有可能性：

如果 $a > b$ ，输出  $b \ a$  否则 输出  $a \ b$ ；

### 思路2：

如果 $a > b$ ，交换 $a$ 、 $b$ 的值实现排序并输出：

```
**核心步骤**
```

```
t=a;
```

```
a=b;
```

```
b=t;
```

## 2. 三个数的排序

### 题目描述

输入三个数，按由大到小顺序打印出来。

### 输入

输入只有一行，包括3个整数。之间用一个空格分开。

### 输出

输出只有一行，包括3个整数。之间用一个空格分开。

### 样例输入

```
3 8 2
```

### 样例输出

```
8 3 2
```

**思路：**采用排队法，即假设操场上站了一排同学，希望他们按照从矮到高（由小到大）排好队，大家只需按照“向后看，如果后面的同学比你高，你俩就换位置”的思路，逐轮排序即可实现。

### 第1轮：

if(第1位身高>第2位){交换1、2两位同学};

```
if(第2位身高>第3位){交换2、3两位同学};
```

第1轮比较结束，你可以得出结论吗？

**注：**每个位置的同学都参加比较了，本轮比较结束

## 第2轮

```
if(第1位身高>第2位){交换1、2两位同学};
```

**问题：如果3个变量a、b、c排序该如何实现？**

**注：**变量a、b、c相当于站队的位置，相当于对应存储同学的位置

**采用排队法，使得a b c由小到大排序：**

```
a = 3    b = 2    c = 3;  
if (a > b), 交换ab的值，使得 a = 2 , b = 3 , c = 1;  
if (b > c), 交换bc的值，使得 a = 2 , b = 1 , c = 3;//此时可确定c为最大值  
if (a > b), 交换ab的值，使得 a = 1 , b = 2 , c = 3;//结果有序
```

## 4.课堂案例

### 1. 求任意三位数打乱次序后组成最大值

#### 题目描述

任意输入一个三位整数，再把它的次序打乱重新组合一个新的三位整数，使其值最大。

#### 输入

输入只有一行，包括1个整数。

#### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

#### 样例输入

```
470
```

#### 样例输出

```
740
```

### 2. 判断三个整数是否相邻

#### 题目描述

判断三个整数是否相邻，是输出"TRUE"，否则输出"FALSE"。

#### 输入

输入只有一行，包括3个整数。

#### 输出

输出只有一行。

#### 样例输入

```
1 3 2
```

#### 样例输出

TRUE

### 提示

三个整数不一定是有序的，例如：1 3 2，是相邻的数！

### 3. 三角形的类别

#### 题目描述

输入三个整数，以这三个数为边长，判断是否构成三角形；若不能输出"no",若构成三角形，进一步判断它们构的是：锐角三角形或直角三角形或钝角三角形.分别输出"ruijiao","zhijiao","dunjiao"

#### 输入

三个整数

#### 输出

一个字符串

#### 样例输入

```
3 4 5
```

#### 样例输出

```
zhijiao
```

### 提示

两个短边的平方和等于一个长边的平方时为直角三角形 两个短边的平方和小于一个长边的平方时为钝角三角形 两个短边的平方和大于一个长边的平方时为锐角三角形

## 5.课后作业

### 1. 求四个数的最大值

#### 题目描述

已知有四个不等的数，将其中的最大数找出来。

#### 输入

输入只有一行，包括4个整数。之间用一个空格分开。

#### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

#### 样例输入

```
1 9 8 6
```

#### 样例输出

```
9
```

**注：** 可采用“打擂法”

### 2. 判断数据能否构成三角形

### 题目描述

判定三条线段a, b, c能否构成一个直角三角形。如果能构成, 请计算出面积 (保留一位小数) , 不然输出'No'。

### 输入

一行, 三个整数。

### 输出

面积 (保留一位小数) 或 No

### 样例输入

```
3 4 5
```

### 样例输出

```
6.0
```

## 3. 做纸箱最少需要多少面积的硬纸板?

### 题目描述

请问做一个尺寸为 $a * b * c$  (单位: 厘米) 的开口的立方体纸箱 (只有一个面是不需要封的, 其余5个面都需要封起来, 这样算开口的), 最少需要多少平方厘米的纸。 (6.1.6)

### 输入

三个整数: a b c

### 输出

制作该纸箱需要的最少硬纸板的面积, 一个整数

### 样例输入

```
5 3 4
```

### 样例输出

```
74
```

## 第八章 while循环

### 1.什么是循环

1. 入门问题: 请输出1~10000之间的每个数

2. 什么是while循环?

while是计算机的一种基本循环模式。当满足条件时进入循环。进入循环后当条件不满足时, 执行完循环体内全部内容后再跳出。

3. while循环的基本结构:

```
while(循环条件){  
    当条件成立时执行;  
    当条件成立时执行;  
    当条件成立时执行;  
    . . .  
}
```

注：： 循环是当条件满足时就执行语句，直到条件不满足

### 对比if

```
if(判断条件){  
    如果条件成立就执行;  
}
```

注： if是如果条件满足就执行语句，但只执行一次就结束

#### 4. 案例：请写出程序的输出结果：

##### 程序1：

```
int x = 1;  
if(x <= 10){  
    cout<<x<<endl;  
}
```

以上程序输出的结果是：

注： 1

##### 程序2：

```
int x = 1;  
while(x <= 10){  
    cout<<x<<endl;  
}
```

以上程序输出的结果是：

注： 死循环，一直输出1

## 2.while的基本应用

### 1. 课堂案例

请使用while循环输出1~10的每个数

```
#include<iostream>  
using namespace std;  
int main(){  
    //循环的初始值  
    int i=1;  
    //循环条件  
    while(i <= 10){  
        //输出数字  
        cout<<i<<endl;  
        //改变条件  
    }
```

```
i = i+1;  
}  
return 0;  
}
```

### 注意：

1. 到了循环阶段需要学会跟着程序的思维逐步分析每一步得到的结果，可以找一张草稿纸记录变量变化的结果；
2. 注意掌握while循环三要素：**初始值、循环条件**（满足时才会循环）、**让循环终止的语句**（本案例中是让  $i > 10$  来终止循环）；
3.  $i++$  的含义等同于： $i = i + 1$ ；
4. 请回答一下问题，这三个问题对于理解while循环至关重要，一定要理解为何会有这种结果：

1. 如果程序没有  $i++$  会输出什么结果？

答：死循环，因为  $i$  值不变则会一直满足循环条件

2. 如果while循环结束，输出  $i$  的值也就是类似如下程序，会输出多少？为什么？

```
int i;  
i=1;  
while(i<=10){  
    cout<<i<<endl;  
    i++;  
}  
//在这里输出i的值，会输出几?  
cout << "循环结束, i="<<i<<endl;
```

答：循环结束后， $i$  的值是 11。

3. 如果将循环内部的  $cout << i << endl$  和  $i++$  位置互换，会输出什么结果？为什么？

```
int i;  
i=1;  
while(i<=10){  
    i++;  
    cout<<i<<endl;  
}
```

答：输出 2~11。

**注意：理解了问题iii就需要注意， $i++$  的位置一定不要随意放置，不同的位置将会让程序产生不一样的结果**

**补充：变量名尽量见名知意**

1.  $s ==> sum$  (总和)
2.  $c ==> count$  (个数)
3.  $n ==> number$  (数字)
4.  $a ==> average$  (平均)

## 2. 课堂练习

1. 请输出 10~1 之间的所有数字

```
#include<iostream>
using namespace std;
int main(){
    int i = 9;
    while(i >= 1){
        cout<<i<<endl;
        i--;//相当于i = i-1;
    }
    return 0;
}
```

## 2. 请输出50~100之间的所有奇数

1. 解法1：取第一个奇数然后递增2

```
#include<iostream>
using namespace std;
int main(){
    int i = 51;
    while(i <= 100){
        cout<<i<<endl;
        i = i + 2;
    }
    return 0;
}
```

2. 解法2：循环50~100之间所有数字并逐个判断是否为奇数，如果是就输出

```
#include<iostream>
using namespace std;
int main(){
    int i = 50;
    while(i <= 100){
        if(i%2 != 0){
            cout<<i<<endl;
        }
        i++;
    }
    return 0;
}
```

## 3. 请输出1~n（n需要从键盘获取）之间所有是2的倍数但不是3的倍数的数字，由小到大输出

1. 第一步：输出1~n之间的所有数字

```

#include<iostream>
using namespace std;
int main(){
    int n ,i = 1;
    cin >> n;
    while(i <= n){
        cout<<i<<endl;
        i++;
    }
    return 0;
}

```

## 2. 第二步：判断是否满足条件，满足的话就输出

```

#include<iostream>
using namespace std;
int main(){
    int n ,i = 1;
    cin >> n;
    while(i <= n){
        if(i % 2 == 0 && i % 3 != 0){
            cout<<i<<endl;
        }
        i++;
    }
    return 0;
}

```

**注意：**需要学会将一个问题拆解成简单的问题来解决，保证每个步骤都是正确的，则结果一定就是正确的。

## 4. 请输出1~n（n需要从键盘获取）之间所有是2的倍数但不是3的倍数的数字的个数，总和是多少？（分步骤解决）

### 1. 第一步：输出1~n之间的每一个数字

```

#include<iostream>
using namespace std;
int main(){
/*
第一步：输出所有1~n之间的数字
第二步：判断是否满足条件
第三步：删除（注释掉）输出语句，在输出语句处实现计数和求和
*/
    int n ,i = 1;
    cin >> n;
    while(i <= n){
        cout<<i<<endl;
        i++;
    }
    return 0;
}

```

### 2. 判断是否满足题目要求：

```

#include<iostream>
using namespace std;
int main(){
    /*
    第一步：输出所有1~n之间的数字
    第二步：判断是否满足条件
    第三步：删除（注释掉）输出语句，在输出语句处实现计数和求和
    */
    int n ,i = 1;
    cin >> n;
    while(i <= n){
        if(i % 2 == 0 && i % 3 != 0){
            cout<<i<<endl;
        }
        i++;
    }
    return 0;
}

```

3. 第三步，在输出位置实现求和和计数，在循环结束后输出结果：

```

#include<iostream>
using namespace std;
int main(){
    /*
    第一步：输出所有1~n之间的数字
    第二步：判断是否满足条件
    第三步：删除（注释掉）输出语句，在输出语句处实现计数和求和
    */
    //s为总数，c为个数，需要给初始值0,
    //不然会出现很奇怪的数字
    int n ,i = 1 , s = 0 , c = 0;
    cin >> n;
    while(i <= n){
        if(i % 2 == 0 && i % 3 != 0){
            //cout<<i<<endl;
            c++;
            s = s + i ;
        }
        i++;
    }
    cout << c << endl;//输出个数
    cout << s << endl;//输出总数
    return 0;
}

```

**总结：**

1. 关于如何计数：每遇到一个满足条件的数字就让变量c自加一； **(注意初始值应为0)**
2. 关于如何求和：每遇到一个满足条件的数字就让数字叠加到变量s上。 **(注意初始值应为0)**

**注意：**

1. 学会求和以及计数

2. 学会将问题拆解为若干步骤（这个很重要），确保每一步都正确则最终结果一定是正确的，以本题为例：

第一步：输出所有1~n之间的数字

第二步：判断是否满足条件

第三步：删除（注：释掉）输出语句，在输出语句处实现计数和求和

## 3.课堂练习

### 1. 编程求 $1*2*3*...*n$

#### 题目描述

编程求 $1*2*3*...*n$

#### 输入

输入一行，只有一个整数n( $1 \leq n \leq 10$ )

#### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

#### 样例输入

5

#### 样例输出

120

#### 思路：

1. 循环输出1~n的每个数
2. 求出他们的乘积

```
#include<iostream>
using namespace std;
int main(){
    int i=1,s=1,n;
    cin>>n;
    while(i<=n){
        //cout<<i<<endl;
        s=s*i;
        i++;
    }
    cout<<s<<endl;
    return 0;
}
```

#### 注意：

1. 要分清输出是在循环中还是循环结束
  2.  $i++$ 、 $i--$ 与 $i=i+1$ 、 $i=i-1$ 的异同与应用
2. 所有不超过1000的数中含有数字3的自然数

#### 题目描述

编程求出所有不超过1000的数中，含有数字3的自然数，并统计总数。

**输入**

无

**输出**

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

**思路：**

1. 循环1~999之间的每个数

```
int i=1;
while(i<10000){
    cout<<i<<endl;
    i++;
}
```

2. 拆分每个数字的每一位，判断是否含有3，有3就输出

```
int i=1,g,s,b;
while(i<10000){
    b = i / 100;
    g = i % 10;
    s = i / 10 % 10;
    if(g==3 || s==3 || b==3){
        cout<<i<<endl;
    }
    i++;
}
```

3. 计数（统计共有多少个这样的i）

```
int i=1,g,s,b,c = 0;//c用来计数，从0开始
while(i<10000){
    b = i / 100;
    g = i % 10;
    s = i / 10 % 10;
    if(g==3 || s==3 || b==3){
        //cout<<i<<endl;
        c++;
    }
    i++;
}
cout<<c<<endl;
```

### 3. 求数字

**题目描述**

输出1—999中有因数3，且至少有一位数字是5的数

**输入**

无

**输出**

若干个数 每行一个

### 思路：

1. 输出1~999；
2. 判断是否满足条件，满足再输出

```
int i=1,g,s,b;
while(i<10000){
    b = i / 100;
    g = i % 10;
    s = i / 10 % 10;
    if(i % 3 == 0 && (g == 5 || s == 5 || b == 5)){
        cout<<i<<endl;
    }
    i++;
}
```

**注意：&&（与）的优先级高于||（或）**

### 4. 求满足条件的整数个数

#### 题目描述

在1 - n中，找出能同时满足用3除余2，用5除余3，用7除余2的所有整数的个数，如果没有请输出0。

#### 输入

输入一行，只有一个整数n(1<=n<=2000)

#### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

#### 样例输入

100

#### 样例输出

1

### 思路：

1. 输出1~n的每个数
2. 判断是否满足条件 ( $i \% 3 = 2, i \% 5 = 3, i \% 7 = 2$ )，满足则输出
3. 求出满足条件的数字的个数（计数）

## 4. 课后作业

### 1. 编程求解 $1+1/2+1/3+\dots+1/n$

#### 题目描述

编程求 $1+1/2+1/3+\dots+1/n$

#### 输入

输入一行，只有一个整数n(1<=n<=200)

#### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个实数。（保留3位小数）

### 样例输入

```
5
```

### 样例输出

```
2.283
```

## 2. 求100到999范围内所有水仙花数

### 题目描述

所谓水仙花数，就是指各位数字立方之和等于该数的数； $a^3$

称为a的立方，即等于 $a \times a \times a$ 的值。例如：因为 $153 = 1^3 + 5^3 + 3^3$ ，所以153是一个水仙花数。

### 输入

无

### 输出

若干行，每行一个整数，表示该范围内的所有水仙花数。按从小到大的顺序输出。

## 3. 编程求1平方+2平方+...+n平方

### 题目描述

编程求1平方+2平方+...+n平方

### 输入

输入一行，只有一个整数n( $1 \leq n \leq 200$ )

### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

### 样例输入

```
5
```

### 样例输出

```
55
```

## 4. 编程求1+3+5+...+n

### 题目描述

编程求 $1+3+5+\dots+n$

### 输入

输入一行，只有一个整数n ( $1 \leq n \leq 9999$ ) 这里n为奇数。

### 输出

输出只有一行

### 样例输入

```
99
```

### 样例输出

2500

## 5. 编程求解 $1+2+3+\dots+n$

### 题目描述

编程求解下列式子的值：  $S=1+2+3+\dots+n$

### 输入

输入一行，只有一个整数n( $1 \leq n \leq 1000$ )

### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

### 样例输入

100

### 样例输出

5050

## 6. 求 $100+97+\dots+4+1$ 的值。

### 题目描述

求 $100+97+\dots+4+1$ 的值

### 输入

无

### 输出

输出一行，即求到的和。

## 7. 编程求 $1*2*3*\dots*n$

### 题目描述

编程求 $1*2*3*\dots*n$

### 输入

输入一行，只有一个整数n( $1 \leq n \leq 10$ )

### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

### 样例输入

5

### 样例输出

120

## 8. 求数II

### 题目描述

在1—500中，找出能同时满足用3除余2，用5除余3，用7除余2的所有整数

**输入**

无

**输出**

若干个数

每行一个

## 9. 寻找雷劈数

**题目描述**

把整数3025从中剪开分为30和25两个数，此时再将这两数之和平方，计算结果又等于原数。

$(30+25)*(30+25)=55*55=3025$ ，这样的数叫“雷劈数”。

求所有符合这样条件的四位数。 $(ab+cd)*(ab+cd)=abcd$

**输入**

无

**输出**

若干行，每行一个雷劈数，从小到大输出。

# 第九章 for循环

## 1.什么是for循环

### 1. for循环的基本结构及与while的异同

输出从m到n之间所有数的while实现方法中有三个组成部分：**初始值、循环条件、让循环终止的语句**，如循环1~10：

```
#include<iostream>
using namespace std;
int main(){
    int i=1;
    while(i<11){
        cout<<i<<endl;
        i++;
    }
}
```

为了方便表达，我们将三个组成部分放在一起表达，即**for循环**：

```
for(初始值；循环条件；让循环停止的方法){
    循环语句;
}
```

**应用**

```
int i;
for(i = 1 ; i < 11 ; i++){
    cout<<i<<endl;
}
```

**或者**

```
for(int i = 1 ; i < 11 ; i++){
    cout<<i<<endl;
}
```

## 2. 理解for循环的执行顺序

```
for(语句1; 语句2; 语句3){
    语句4;
}
```

优先执行语句1且只执行一次；

然后循环执行：语句2=>语句4=>语句3=>语句2=>语句4=>语句3.....直到语句2不成立终止

注意理解for循环的执行顺序：

1. 按照上面知识点中关于语句执行过程的讲解，可以看出首先执行`i = 1`，为`i`赋初始值；
2. 接下来判断`i < 11`是否成立；
3. 如果成立，则接下来执行`cout<<i<<endl`
4. 执行`i++`,然后继续判断`i < 11`是否成立
5. 特别注意：循环结束后`i`的值是多少？

```
for(int i = 1 ; i < 11 ; i ++){
    cout << i << endl;
}
cout << "循环结束, i = " << i << endl;
```

6. 理论上for循环中分号隔开的3条语句可以都没有，但2个分号需要保留（但在实际的编程中虽然语法正确，但不容易理解，因此不建议使用），例子：

```
int i = 1;
for( ; i < 11 ; ){
    cout << i << endl;
    i++;
}
```

## 2. for循环的案例

### 1. 4位反序数

#### 题目描述

设N是一个四位数，它的9倍恰好是其反序数，求N。反序数就是将整数的数字倒过来形成的整数。例如：1234的反序数是4321。

#### 输入

无

#### 输出

输出N这个四位数

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

```

int main(){
/*
思路:
1.循环所有4位数
2.求出每个4位数的反序数
(拆出每个数位上的数并反序)
3.判断是否为该数字的9倍,若是就输出
*/
int g,s,b,q,x;
for(int i = 1000 ; i < 10000 ; i ++){
    q = i / 1000 ;
    b = i / 100 % 10;
    s = i / 10 % 10;
    g = i % 10;
    x = g * 1000 + s * 100 + b * 10 +q;
    if(x / i == 9){
        cout << i << endl;
    }
}
return 0;
}

```

## 2. 寻找雷劈数

### 题目描述

把整数3025从中剪开分为30和25两个数，此时再将这两数之和平方，计算结果又等于原数。

$(30+25) \times (30+25) = 55 \times 55 = 3025$ ，这样的数叫“雷劈数”。

求所有符合这样条件的四位数。 $(ab+cd) \times (ab+cd) = abcd$

### 输入

无

### 输出

若干行，每行一个雷劈数，从小到大输出。

```

#include<iostream>
using namespace std;
int main(){
/*
思路:
1.循环出1000~9999之间所有数;
2.拆出每个数的前两位和后两位
    i = 3025
    前两位 a = i / 100;
    后两位 b = i % 100;
3.判断是否满足条件 (a + b) * (a + b) == i;
*/
int a,b;
for(int i = 1000 ; i < 10000 ; i++){
    a = i / 100;
    b = i % 100;
    if((a + b) * (a + b) == i){
        cout << i << endl;
    }
}
return 0;
}

```

```
}
```

### 3. 能被5整除且至少有一位数字是5的所有整数的个数

#### 题目描述

找出1 - N中能被5整除且至少有一位数字是5的所有整数的个数.N<32767

#### 输入

输入只有一行，只有1个整数N。

#### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

#### 样例输入

```
9999
```

#### 样例输出

```
1271
```

```
#include<iostream>
using namespace std;
int main(){
    /*
    思路：
    1. 循环出1~n中所有数
    2. 判断是否为5的倍数且有一位数位是5，如果是，则输出
    3. 计数
    */
    int g,s,b,q,w,c = 0,n;
    cin >> n;
    for(int i = 1 ; i <= n ; i ++){
        w = i / 10000;
        q = i / 1000 % 10;
        b = i / 100 % 10;
        s = i / 10 % 10;
        g = i % 10;
        if(i % 5 == 0 && ( w == 5 || q == 5 || b == 5 || s == 5 || g == 5 )){
            c++;
        }
    }
    cout << c << endl ;
    return 0;
}
```

注意：`&&` 的优先级高于 `||`

## 3. for循环和while的使用场景

1. for：如果能够明确循环的范围是n~m，for循环更加便捷；
2. while：如没有明确范围但是有循环条件，则while更适用

例子：请问一个正整数能够整除几次2？

## 题目描述

请问一个正整数n能够整除几次2?

比如: 4可以整除2次2, 100可以整除2次2, 9可以整除0次2。

## 输入

从键盘读入一个正整数n

## 输出

输出一个整数, 代表n能够整除2的次数

## 样例输入

```
8
```

## 样例输出

```
3
```

```
#include<iostream>
using namespace std;
int main(){
    /*
    思路:
    当n能够整除2时, 计数器自增1且n=n/2
    */
    int n,c=0;
    while(n % 2 == 0){
        c++;
        n = n / 2 ;
    }
    cout << c ;
    return 0;
}
```

## 4.课堂练习

### 1. 求恰好使 $s=1+1/2+1/3+\dots+1/n$ 的值大于X时n的值

#### 题目描述

求恰好使 $s=1+1/2+1/3+\dots+1/n$ 的值大于X时n的值。( $2 \leq X \leq 10$ )

#### 输入

输入只有一行, 包括1个整数X。

#### 输出

输出只有一行 (这意味着末尾有一个回车符号) , 包括1个整数。

#### 样例输入

```
2
```

#### 样例输出

**提示：**

1. 循环停止条件： $s > x$ ;
2. 循环条件： $s \leq x$ ;
3.  $s = 0$ ;
4. `while(s <= x){}`

**源代码**

```
#include<iostream>
using namespace std;
int main(){
    double s = 0 ;
    int x , i = 1;
    cin >> x;
    while(s <= x){
        s = s + 1.0 / i ;
        i++;
    }
    cout << i - 1 << endl;
    return 0;
}
```

**2. 韩信点兵****题目描述**

韩信有一对士兵，他想知道有多少人，他就让士兵报数，如果按照1到5报数，最末一个士兵报的数为1；按照1到6报数，最末一个士兵报的数为5；按照1到7报数，最末一个士兵报的数为4；最后再按1到11报数，最末一个士兵报的数为10，请问韩信这队士兵最少有多少人？

**输入**

无

**输出**

输出这队士兵最少有多少人

**思路：**

1. 从1开始找这样一个数，满足： $i \% 5 == 1 \ \&\& i \% 6 == 5 \ \&\& i \% 7 == 4 \ \&\& i \% 11 == 10$ ，若不满足要求则继续寻找下一个
2. 上述整理的条件是一个循环停止条件：满足该条件即可停止循环
3. 反过来说，循环条件应该是 $i \% 5 != 1 \ \mid\mid i \% 6 != 5 \ \mid\mid i \% 7 != 4 \ \mid\mid i \% 11 != 10$

```
#include<iostream>
using namespace std;
int main(){
    int i=1;
    while(i % 5 != 1 || i & 6 != 5 || i % 7 != 4 || i % 11 != 10){
        i++;
    }
    cout << i ;
}
```

### 3. 求各位数字之和

#### 题目描述

输入一个正整数N ( $0 \leq N \leq 2147483647$ )，求它的各位数字之和。

#### 输入

一行，一个正整数N。

#### 输出

一行，一个整数。

#### 样例输入

```
189
```

#### 样例输出

```
18
```

#### 思路：

难点在于无法获知输入的数字是几位数，所以可以采用while用短除法，可以从个位开始取，然后去掉取过的个位，直到取到0，具体方法为：用  $n \% 10$  取到个位， $n / 10$  去掉已经取过的个位，如此循环到0即可。

```
#include<iostream>
using namespace std;
int main(){
    int n,s=0;
    cin >> n;
    while(n != 0){
        s += n%10 ;//s+=n 等效于 s = s + n;
        n /= 10;
    }
    cout<< s << endl;
    return 0;
}
```

### 4. “小黄数”

#### 题目描述

小黄发明了一种数字游戏，输入一个正整数N ( $0 < N < 2147483647$ )，将这个数倒着合成一个新数后输出，这个数字叫做“小黄数”。

比如：543，倒过来是345（请注意：34500，倒过来是543，不是00543）！

### 输入

一行，一个正整数N。

### 输出

一行，一个类型为正整数的“小黄数”。

### 样例输入

```
345
```

### 样例输出

```
543
```

```
#include<iostream>
using namespace std;
int main(){
    int n , s = 0;
    cin >> n ;
    while(n != 0){
        s = s * 10 + n % 10 ;
        n /= 10 ;
    }
    cout << s ;
    return 0;
}
```

## 5.课后作业

### 1. 回文偶数

#### 题目描述

小明发现有一类数非常有趣，他们正过来读和反过来读是一样的，比如：121、202、383等，小明给这类数起了一个名字，叫做回文数。

请你写程序帮助小明找出所有3位的既是回文数，又是偶数的数，比如：202就是满足条件的数，而121虽然是回文数但不是偶数。

#### 输入

无

#### 输出

所有满足条件的3位的回文偶数，每行1个。

### 2. 同因查找

#### 题目描述

求出10至1000之内能同时被2、3、7整除的数，并输出。每行一个。

#### 输入

无

#### 输出

按要求输出满足条件的数，每行1个

### 3. 与7无关的数

### 题目描述

一个整数，如果这个数能够被7整除，或者其中有一位是7，我们称为这个数是与7有关的数。比如：14能被7整除，17有一位为7，这两个数都是与7有关的数。

请你编程求出1~n (n<=999) 中，与7无关的数的总和是多少？

比如1~10中与7无关的数的和为：1+2+3+4+5+6+8+9+10=48。

### 输入

一个整数n (n<=999)

### 输出

1~n中与7无关的数的总和

### 样例输入

```
10
```

### 样例输出

```
48
```

## 4. 奇数及偶数和

### 题目描述

给出一个正整数n (1≤n≤1000)，求出1, 2, .....n中全部奇数和以及全部偶数的和。

例如：n=9

奇数和 1+3+5+7+9=25

偶数和 2+4+6+8=20

### 输入

一个整数n。

### 输出

一行，奇数和与偶数和，中间一个" " (空格)。

### 样例输入

```
6
```

### 样例输出

```
9 12
```

## 5. 小丽找数

### 题目描述

小丽同学想在1~n中找出这样的数，这个数的各个位的和不能被2整除也不能被5整除，比如3、12、25、30、100。这些数都满足各个位的和不能被2和5整除。

请你编程找出1~n中这些数有多少个？

### 输入

一个整数n (n<=9999)

### 输出

1~n中满足条件的数的个数

### 样例输入

```
50
```

### 样例输出

```
20
```

## 6. 角谷猜想

### 题目描述

日本一位中学生发现一个奇妙的定理，请角谷教授证明，而教授无能为力，于是产生了角谷猜想。猜想的内容：任给一个自然数，若为偶数则除以2，若为奇数则乘3加1，得到一个新的自然数后按上面的法则继续演算。若干次后得到的结果必为1。请编写代码验证该猜想：求经过多少次运算可得到自然数1。如：输入22，

```
22/2=11  
11*3+1=34  
34/2=17  
17*3+1=52  
52/2=26  
26/2=13  
13*3+1=40  
40/2=20  
20/2=10  
10/2=5  
5*3+1=16  
16/2=8  
8/2=4  
4/2=2  
2/2=1
```

经过15次运算得到自然数1。

### 输入

一行，一个正整数n。 (1 <= n <= 20000 )

### 输出

一行，一个整数，表示得到1所用的运算次数。

### 样例输入

```
22
```

### 样例输出

```
15
```

## 7. 爱因斯坦的数学题

### 题目描述

爱因斯坦出了一道这样的数学题：有一条长阶梯，若每步跨2阶，则最最后剩一阶，若每步跨3阶，则最后剩2阶，若每步跨5阶，则最后剩4阶，若每步跨6阶则最后剩5阶。只有每次跨7阶，最后才正好一阶不剩。请问这条阶梯最少共有多少阶？

**输入**

无

**输出**

这条阶梯最少的阶数

## 第十章 程序的报错与调试

### 1.程序错误的常见类型:

1. **语法错误**: 程序语法有误, 运行报错 (红叉)
2. **逻辑错误**: 语法没有问题, 可以运行, 但是结果错误

### 2.常见的语法错误:

案例:求一个三位数各个数位上的数字之和

```
int n , g , s , b , s ;  
cin >> n ;  
b = n / 100 ;  
s = n / 10 % 10  
g = n % 10;  
cout<< b + s + g << endl ;
```

程序报错	错误解析
[Error] redeclaration of 'int s'	重复定义"s"
[Error] expected ';' before 'g'	在g之前缺少分号
[Error] 'S' was not declared in this scope	变量S未定义

### 3.逻辑错误

题目: 能被5整除且至少有一位数字是5的所有整数的个数

#### 题目描述

找出1 - N中能被5整除且至少有一位数字是5的所有整数的个数.N<32767

#### 输入

输入只有一行, 只有1个整数N。

#### 输出

输出只有一行 (这意味着末尾有一个回车符号), 包括1个整数。

#### 样例输入

```
9999
```

#### 样例输出

```
1271
```

## 示例代码（错误）：

```
#include<iostream>
using namespace std;
int main(){
    int n,g,s,b,q,w,c;
    cin>>n;
    for(int i=1;i<n;i++){
        g = i % 10 ;
        s = i / 10 % 10;
        b = i % 100 / 10;
        q = i / 1000 % 10;
        w = i / 10000;
        if(i % 5 == 0 && g == 5 || s == 5 || b == 5 || q == 5 || w == 5){
            c += 1;
        }
        cout << c << endl;
    }
    return 0;
}
```

上述代码肯定是有错误的，但是我们应该如何正确地修正呢？一定要养成良好的代码调试习惯。其实有点类似解题的思路：

1. 循环1~n是否有误？
2. 拆出i各个数位上的数字是否有误？
3. 判断条件是否有问题？
4. 计数是否有误？
5. 输出是否有误？

## 3.输出调试法

1. 选中无需测试的代码并按下“Ctrl+?”注：释掉；
2. cout输出想要观察验证的结果
3. 可以适当修改参数以适应调试测试需求

## 4.单步调试

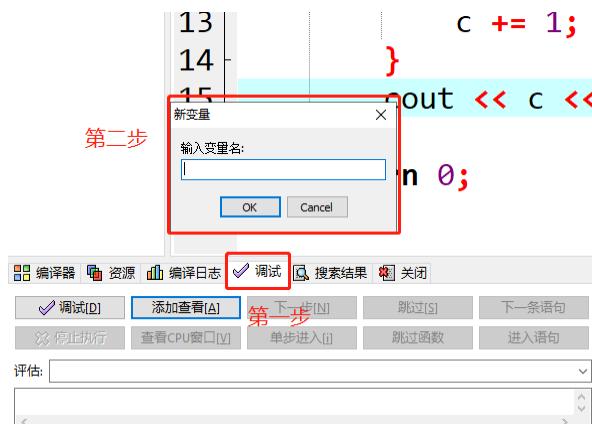
1. 设置断点（点击对应的行号即可），一般在cin之后

```

1 #include<iostream>
2 using namespace std;
3 int main(){
4     int n,g,s,b,q,w,c;
5     cin>>n;
6     for(int i=1;i<n;i++){
7         g = i % 10 ;
8         s = i / 10 % 10;
9         b = i % 100 / 10;
10        q = i / 1000 % 10;
11        w = i / 10000;
12        if(i%5==0&&g==5||s==5||b==5||q==5||w==5){
13            c += 1;
14        }
15        cout << c << endl;
16    }
17    return 0;
18 }

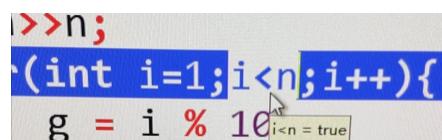
```

2. 在程序下方的“调试”中输入需要查看观察的变量：



3. 点击“调试” (上方和下方都有紫色√)，然后下一步，关注：运行结果与预期是否一致以调试和纠错

\*\* 补充\*\*执行到断点时会对应变成蓝色，选中对应变量、表达式可以看到当前的值



需要注意的是蓝色的时候表示这一行还没有执行，只是到了这一行，需要点“下一步”才会执行并跳到下一行

## 5.课堂练习

按照上述两种方法将程序源代码修改成正确的

# 第十一章 嵌套循环

## 1.什么是嵌套循环

### 1. 字符图形 1-星号矩形

#### 题目描述

打印字符图形。输出n行n列"\*\*"

#### 输入

一个整数 n(0 < n < 10)

输出

一个矩形字符图形

### 样例输入

```
3
```

### 样例输出

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

**思路：打印n行，每行n个**

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

### 注意：

1. 通过逐步分析，理解程序的运行、输出顺序。可将内部循环看成一个整体，即输出一行\*，输出完成后换一行，将两个功能看做一个整体，循环n次
2. 可以通过单步调试进一步加深理解

## 2. 加深理解

读下面的程序，计算出程序输出的结果

1. 阅读程序：

```
int i,n;  
n = 3;  
for(i = 1;i <= n;i++){  
    cout<<i;  
}  
cout<<endl;  
for(i = 1;i <= n;i++){  
    cout<<i;  
}  
cout<<endl;
```

**结果:**

```
123
```

```
123
```

**注意: 如果是非嵌套循环, 可以使用同名变量, 可参考第二章第二节第1小节第5点笔记**

2. 阅读程序

```
int i,j,n;
n = 3;
for(i = 1;i <= n;i++){
    for(i = 1;i <= n;i++){
        cout << "*";
    }
    cout << endl;
}
```

**结果:**

```
***
```

**注意: 嵌套循环是不可以使用同名变量的, 这样会导致内外循环共用一个变量, 即内部循环可以修改外部循环使用的变量的值, 从而达不到嵌套循环的要求**

3. 阅读程序

```
int i,j,n;
n = 3;
for(i = 1;i <= n;i++){
    for(j = 1;j <= n;j++){
        cout << i;
    }
    cout << endl;
}
```

**结果:**

```
111
```

```
222
```

```
333
```

4. 阅读程序

```
int i,j,n;
n = 3;
for(i = 1;i <= n;i++){
    for(j = 1;j <= n;j++){
        cout << j;
    }
    cout << endl;
}
```

结果：

```
123  
123  
123
```

5. 阅读程序

```
int i,j,n;  
n = 3;  
for(i = 1;i <= n;i++){  
    for(j = 1;j <= i;j ++){  
        cout << j;  
    }  
    cout << endl;  
}
```

```
1  
12  
123
```

6. 阅读程序

```
int i,j,n;  
n = 3;  
for(i = 1;i <= n;i++){  
    for(j = 1;j <= i;j ++){  
        cout << j*i;  
    }  
    cout << endl;  
}
```

```
1  
24  
369
```

### 3. 嵌套循环的基本结构

```
for(i 初始值;i 循环条件;i 让循环停止的方法){  
    for(j 初始值;j 循环条件;j 让循环停止的方法){  
    }  
}
```

注意：外层和内层不可以使用同一个变量作为循环控制

### 4. 嵌套循环的图形题例题库1

#### 1. 数字矩形(1)

题目描述

从键盘读入一个整数n，输出如下图形

如：n=3，输出

111

222

333

如：n = 5，输出

11111

22222

33333

44444

55555

### 输入

一个整数n (n<10)

### 输出

输出n行的图形

### 样例输入

```
3
```

### 样例输出

```
111
```

```
222
```

```
333
```

### 示例代码

```
int main(){
    /*
    思路：
    输出n行，每行n个数字
    第1行是1，第2行是2.....
    归纳得出第i行是i
    */
    int n;
    cin >> n;
    for(int i = 1;i <= n;i ++){
        for(int j = 1;j <= i;j ++){
            cout << i;
        }
        cout<<endl;
    }
    return 0;
}
```

## 2. 数字矩形(2)

### 题目描述

从键盘读入一个整数n，输出如下图形

如：n=3，输出

123

123

123

如: n = 5, 输出

12345

12345

12345

12345

12345

### 输入

一个整数n (n<=10)

### 输出

输出n行的图形

### 样例输入

```
3
```

### 样例输出

```
123
```

```
123
```

```
123
```

```
#include<iostream>
using namespace std;
int main(){
    /*
    思路:
    输出n行, 每行n个数字
    第1行是1~n, 第2行是1~n.....
    归纳可得: 第i行是1~n
    */
    int n;
    cin >> n;
    for(int i = 1;i <= n;i ++){
        for(int j = 1;j <= i;j ++){
            cout << j;
        }
        cout<<endl;
    }
    return 0;
}
```

## 3. 字符图形2-星号直角

### 题目描述

打印字符图形。

### 输入

一个整数 ( $0 < n < 10$ )

**输出**

一个字符图形

**样例输入**

```
3
```

**样例输出**

```
*
```

```
**
```

```
***
```

```
#include<iostream>
using namespace std;
int main(){
    /*
    思路:
    输出n行, 每行n个*
    第1行是1个*2行是2个*....
    归纳可得: 第n是n个*
    */
    int n;
    cin >> n;
    for(int i = 1;i <= n;i ++){
        for(int j = 1;j <= i;j ++){
            cout << "*";
        }
        cout<<endl;
    }
    return 0;
}
```

#### 4. 字符图形3-平行四边形

**题目描述**

输入一个整数打印字符图形

**输入**

一个整数 ( $0 < N < 10$ )

**输出**

一个字符图形

**样例输入**

```
5
```

**样例输出**

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

```
#include<iostream>
using namespace std;
int main(){
/*
思路:
输出n行的图形
每一行先输出空格, 再输出*
空格: 第一行0个, 第二行1个, 第三行2个...
星星: 每行都是n个
归纳可得: 空格是第i 行有 i-1 个。星星是每行都有 n 个
*/
int n;
cin >> n;
for(int i = 1;i <= n;i ++){
    //第一个循环用来打印空格
    //判断条件也可以写成j<=i-1, 结果是一样的
    for(int j = 1;j < i;j ++){
        cout << " ";
    }
    //第二个循环用来打印*
    for(int j = 1;j <= n;j ++){
        cout << "*";
    }
    cout<<endl;
}
return 0;
}
```

## 5.常见的数值规律

问题：如何通过循环变量i得到常见的数列？

```
for(int i=1;i<=5;i++){
    cout<<_____<<" ";
}
```

请问：上述空格应该填什么能够输出以下数列？

1. 1 2 3 4 5

答： i

2. 2 4 6 8 10

答： i\*2

3. 1 3 5 7 9

答： i\*2-1

4. 5 4 3 2 1

答: 6-i

5. 9 7 5 3 1

答:  $(6-i)*2-1$

总结:

1. 连续自然数: `cout<<i<<" "`;
2. 连续偶数: `cout<<2*i<<" "`;
3. 连续奇数: `cout<<i*2-1<<" "`;
4. 倒叙连续的数: `cout<<6-i<<" "`; (最大数+1减去每一个数)

## 6. 嵌套循环的图形题例题库2

### 1. 字符图形4-星号正三角

#### 题目描述

输入一个整数打印字符图形

#### 输入

一个整数 ( $0 < N < 10$ )

#### 输出

一个字符图形, 例如, 输入3, 则输出图形如下: (为方便统计, □代表空格, ×代表\*)

```
□□×  
□×××  
×××××
```

#### 样例输入

```
3
```

#### 样例输出

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

#### 思路:

1. 第1行2个空格1个\*
2. 第2行1个空格3个\*
3. 第3行0个空格5个\*
4. 第*i*行*n-i*个空格 $2*i-1$ 个\*

```
#include<iostream>  
using namespace std;  
int main(){  
    int n;  
    cin>>n;  
    for(int i=1;i<=n;i++){  
        for(int j=1;j<=n-i;j++) {
```

```
        cout<<" ";
    }
    for(int j=1;j<=2*i-1;j++){
        cout<<"*";
    }
    cout<<endl;
}
return 0;
}
```

## 2. 字符图形6-星号倒三角

### 题目描述

输入一个整数打印字符图形

### 输入

一个整数 ( $0 < N < 10$ )

### 输出

一个字符图形, 如:  $n=3$ , 则输出图形如下:

```
xxxxx
□xxx
□□×
```

### 样例输入

```
3
```

### 样例输出

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

### 思路1:

1. 第1行0个空格5个\*
2. 第2行2个空格3个\*
3. 第3行3个空格1个\*
4. 第*i*行*i*个空格( $n-i$ )\*2+1个\*

```
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=i;j++){
            cout<<" ";
        }
        for(int j=1;j<=(n-i)*2+1;j++){
            cout<<"*";
        }
    }
}
```

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

## 思路2：

正三角倒过来即为倒三角，参考代码：

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

## 7.上下对称及空心类图形

### 1. 字符图形7-星号菱形

#### 题目描述

输入一个整数打印字符图形

#### 输入

一个整数（ $0 < N < 10$ ）

#### 输出

一个字符图形，如输入2，则产生5行的菱形：

```
□□×
□×××
×××××
□×××
□□×
```

#### 样例输入

```
2
```

#### 样例输出

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

### 思路：

上下对称的图形我们一般先打印出一半，另一半直接倒过来得到即可。**需要注意的是要首先得到一半图形**

```
#include<iostream>  
using namespace std;  
int main(){  
    int n;  
    cin>>n;  
    //先得到一个正三角  
    for(int i=1;i<=n;i++){  
        for(int j=1;j<=n-i+1;j++){  
            cout<<" ";  
        }  
        for(int j=1;j<=i*2-1;j++){  
            cout<<"*";  
        }  
        cout<<endl;  
    }  
    //得到中间的对称轴  
    for(int i=1;i<=n*2+1;i++){  
        cout<<"*";  
    }  
    cout<<endl;  
    //颠倒得到倒三角  
    for(int i=n;i>=1;i--){  
        for(int j=1;j<=n-i+1;j++){  
            cout<<" ";  
        }  
        for(int j=1;j<=i*2-1;j++){  
            cout<<"*";  
        }  
        cout<<endl;  
    }  
    return 0;  
}
```

## 2. 放大的箭头

### 题目描述

【入门】放大的箭头

请打印n行的放大的箭头（n一定是一个奇数）

如：输出5行的箭头，输出结果如下，为方便理解，我们用□代表空格，实际输出的时候，请输出空格！

每行有n颗星！

```
*****
```

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

## 输入

n, 代表有n行的图形 (n一定是一个奇数)

## 输出

n行的图形!

### 样例输入

```
5
```

### 样例输出

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

## 思路：

1. 先得到一半的图形
2. 输出一半的图形，颠倒得到另一半
3. 删掉中间多出来的一行

```
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    //让i<n/2而不是i<=2来控制上半个图形行数，少一行
    for(int i=0;i<n/2;i++){
        for(int j=0;j<=i-1;j++){
            cout<<" ";
        }
        for(int j=0;j<n;j++){
            cout<<"*";
        }
        cout<<endl;
    }
    //让i起始从i=n/2开始，到0结束，
    //所以会多一行，刚好凑成整的
    for(int i=n/2;i>=0;i--){
        for(int j=0;j<=i-1;j++){
            cout<<" ";
        }
        for(int j=0;j<n;j++){
            cout<<"*";
        }
        cout<<endl;
    }
}
```

```
    }
    return 0;
}
```

### 3. 打印空心等腰三角形

#### 题目描述

从键盘读入一个整数n，代表等腰三角形的边长，请输出一个边长为n的等腰三角形！

为了方便观察，我们在例子中将空格替换成□，将\*替换为x，请在程序中正常输出空格和\*！

如：n=3则输出

```
□□x  
□x□x  
xxxxx
```

n=5则输出：

```
□□□□x  
□□□x□x  
□□x□□□x  
□x□□□□□x  
xxxxxxxxx
```

#### 输入

3

#### 输出

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

#### 思路：

1. 先打印整个图形（非空心）
2. 打印过程中判断，如果不当前是最后一行，只打印第一个和最后一个\*，其余用空格代替
3. 可以先画出图形再慢慢调整

```
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    for(int i=0;i<n;i++){
        for(int j=0;j<n-i-1;j++){
            cout<<" ";
        }
        if(i!=n-1){
            cout<<"*";
            for(int j=0;j<2*i-1;j++){
                cout<<" ";
            }
        }
    }
}
```

```

        cout<<" ";
    }
    if(i!=0){
        cout<<"*";
    }
} else{
    for(int j=0;j<2*i+1;j++){
        cout<<"*";
    }
}

cout<<endl;
}
return 0;
}

```

## 8.常用的快捷键整理

1. **ctrl+a**:全选
2. **ctrl+c**:复制
3. **ctrl+v**:粘贴
4. **ctrl+s**:保存
5. **ctrl+z**:撤销
6. **ctrl+y**:回退

## 9.课后作业

### 1. 字符图形5-星号梯形

#### 题目描述

输入一个整数打印字符图形 N=3 输出

```

□□×××
□×××××
×××××××
N=5
□□□□×××
□□□×××××
□□×××××××
□××××××××
×××××××××

```

#### 输入

一个整数 ( 0 < N < 10 )

#### 输出

一个字符图形

#### 样例输入

3

#### 样例输出

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

## 2. 沙漏

### 题目描述

赵老师最近在编一个操作系统，正好编到鼠标的繁忙状态，需要一个沙漏符号，正好大家都在学C++，你的任务就是帮赵老师编一个程序打印一个沙漏符号。

### 输入

一个整数n， 符号的行数（保证n是大于1的奇数）

### 输出

沙漏符号， 使用“\*”打印

### 样例输入

```
5
```

### 样例输出

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

## 3. 蝴蝶结

### 题目描述

请输出n行的蝴蝶结的形状， n一定是一个奇数！

### 输入

一个整数n， 代表图形的行数！

### 输出

n行的图形

### 样例输入

```
9
```

### 样例输出

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

#### 4. 打印n行的完整的蝴蝶结

##### 题目描述

请从键盘读入一个整数n (n是1~10的范围内的奇数) , 打印出如下图所示的n行的完整的蝴蝶结!

比如, n=5, 则打印图形如下:



##### 输入

一个整数n, 代表图形的行数!

##### 输出

n行的图形!

##### 样例输入

```
5
```

##### 样例输出

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

#### 5. 请输出n行的9\*9乘法表

##### 题目描述

请从键盘读入一个整数n, 代表有n行, 输出n行的9\*9乘法表。

比如, 假设n=5, 则输出如下:

```
1*1=1
2*1=2 2*2=4
3*1=3 3*2=6 3*3=9
4*1=4 4*2=8 4*3=12 4*4=16
5*1=5 5*2=10 5*3=15 5*4=20 5*5=25
```

##### 输入

一个整数n (n<=9)

##### 输出

n行的9\*9乘法表

##### 样例输入

```
3
```

##### 样例输出

```
1*1=1  
2*1=2 2*2=4  
3*1=3 3*2=6 3*3=9
```

## 6. 字符图形9-数字正三角

### 题目描述

输入一个整数打印字符图形

### 输入

一个整数 (  $0 < N < 10$  )

### 输出

一个字符图形

### 样例输入

```
3
```

### 样例输出

```
1  
222  
33333
```

## 9.附加题

### 1. 打印星号三角行

#### 题目描述

打印星号三角行.

#### 输入

输入只有一行，包括1个整数N。N代表行数.

#### 输出

输出N行.

#### 样例输入

```
5
```

#### 样例输出

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

### 2. 挑战赛第二题——放大的X

#### 题目描述

请你编程画一个放大的'X'（大写字母）。如3\*3的'X'应如下所示：

```
x x  
x  
x x
```

5\*5的'X'如下所示：

```
x x  
x x  
x  
x x  
x x
```

### 输入

有一个正奇数n ( $3 \leq n \leq 79$ )，表示放大的规格。

### 输出

打印一个规格为n \* n放大的'X'

### 样例输入

```
5
```

### 样例输出

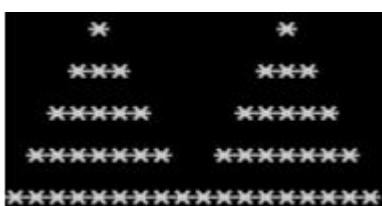
```
x x  
x x  
x  
x x  
x x
```

## 3. 轴对称三角形

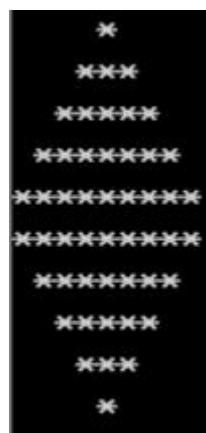
### 题目描述

在数学上，我们发现有一类图形是对称图形。我们对于左右一样的图形叫做沿y轴对称，对于上下一样的图形叫做沿x轴对称。

如下图所示的图形就是沿y轴对称的两个三角形：



而下图所示的图形就是沿x轴对称的两个三角形：



请从键盘读入一个整数n，代表三角形的行数，再读入一个字符，该字符为'x'或者'y'，代表是沿x轴对称还是沿y轴对称，输出对应的轴对称图形。 (8.1)

### 输入

第一样是一个整数n (n <= 10)

第二行是一个字符 (字符'x'或者字符'y')

### 输出

按照要求输出的轴对称图形

#### 样例输入

```
5
```

```
x
```

#### 样例输出

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

## 第十二章 穷举

### 1.什么是穷举?

穷举即列举出所有的可能性并判断出哪些是符合条件的。

### 2.单循环穷举

#### 1. 鸡兔同笼问题

##### 题目描述

鸡兔同笼问题：一个笼子里面有鸡若干只，兔若干只。共有头50个，共有腿160条。求鸡兔各多少只？

##### 输入

##### 输出

两个整数，在一行。

鸡的只数 兔的只数 中间空格隔开！

##### 思路：

循环鸡和兔子可能的所有组合，并判断是否存在满足条件的情况：

```
循环(鸡的只数范围 i = 1~49){
    如果(i 只鸡, 50-i 只兔子的腿数刚好满足160){
        输出鸡和兔子的个数;
    }
}
```

鸡	兔	腿
1	49	$1*2+49*4==160?$
2	48	$2*2+48*4==160?$
3	47	$3*2+47*4==160?$
....	....	....

示例代码：

```
#include<iostream>
using namespace std;
int main(){
    for(int i = 1;i < 50;i ++){
        if(i * 2 + (50 - i) * 4 == 160){
            cout << i << " " << 50-i << endl;
        }
    }
    return 0;
}
```

## 2. 买公园门票

### 题目描述

某公园门票价格为：成人票8元/张，儿童票3元/张；某旅游团来公园游玩，该团内有成人和儿童（成人和儿童都有），共花了40元买门票，请你分别计算出成人和儿童可能的人数，按照成人从少到多，儿童从多到少的规律数出结果。

### 输入

无

### 输出

若干行，每行2个整数用空格隔开，分别代表成人和儿童可能的人数（成人从少到多，儿童从多到少）

### 思路：

循环（穷举）成人可能的人数范围。在有*i*个成人的情况下，买完成人票剩余的金额如果是儿童票价的倍数则说明该组数据符合条件。

成人人数	剩余金额	方案是否可行？
1	$40-1*8=36$	32不是3的倍数，不可行
2	$40-2*8=24$	24是3的倍数，可行
....	....	....

示例代码：

```

#include<iostream>
using namespace std;
int main(){
    //x表示买了成人票剩下的钱, i表示买了几张成人票
    int x;
    //成人票至少一张, 最多  $(40-3)/8$ 张 (至少有一张儿童票)
    for(int i = 1; i <= (40-3)/8; i ++){
        //计算剩下的钱
        x = 40 - i * 8 ;
        if(x % 3 == 0){
            cout << i << " " << x / 3 << endl;
        }
    }
    return 0;
}

```

### 3. 买小猫小狗

#### 题目描述

某动物饲养中心用X元专款购买小狗(每只A元)和小猫(每只B元)两种小动物。  
要求专款专用,(至少猫狗各一),正好用完?请求出方案的总数。如没有请输出0.

#### 输入

输入一行, 只有三个整数.分别为X,A,B. ( $100 < X < 32768$ ;  $1 \leq A, B \leq 100$ )

#### 输出

输出只有一行 (这意味着末尾有一个回车符号) , 包括1个整数。

#### 样例输入

1700 31 21

#### 样例输出

3

```

#include<iostream>
using namespace std;
int main(){
    /*
    思路:
    共x元, 小狗a元/只, 小猫b元/只,
    y表示买小狗剩下的钱,c表示可行方案个数
    至少一只猫/一只狗, 钱刚好用完
    */
    int x , y , a, b , c = 0;
    cin >> x >> a >> b;
    for(int i = 1;i <= (x - b) / a;i ++){
        y = x - i * a ;
        if(y % b == 0){
            c++;
        }
    }
    cout << c << endl;
    return 0;
}

```

```
}
```

#### 4. 阿凡提的难题

##### 题目描述

阿凡提去集市上买餐具，财主正好在卖餐具，所以准备为难一下阿凡提；财主的餐具有2种：大碗和小碗，财主和阿凡提说，你买我的碗，要花光你带的钱，而且，两种碗都要买，买的两种碗的数量都得是偶数，请你编程帮助阿凡提计算，可以有哪些购买的方案呢？

##### 输入

三个整数，分别代表了阿凡提带的钱的数量，大碗的价格，小碗的价格！

##### 输出

所有的购买方案，一行一个方案，先输出大碗的采购只数，再输出小碗的采购只数！

##### 样例输入

```
100 20 10
```

##### 样例输出

```
2 6  
4 2
```

##### 思路：

1. 钱要花完
2. 两种碗都要有
3. 两种碗的数量都要是偶数

```
#include<iostream>  
using namespace std;  
int main(){  
    //n代表多少钱，x表示买i只大碗后剩下的钱  
    int n , a , b , x , i;  
    cin >> n >> a >> b;  
    for(i = 2;i <= (n - 2 * b) / a; i = i + 2){  
        x = n - i * a;  
        if(x % b == 0 && x / b % 2 == 0){  
            cout << i << " " << x / b << endl;  
        }  
    }  
    return 0;  
}
```

## 3.课后作业

### 1. 植树的人数

##### 题目描述

某班学生分2组参加植树活动，甲组有17人，乙组有25人，后来由于需要，从甲组抽调了部分学生去乙组，结果乙组的人数是甲组的2倍，请问从甲组抽调了多少人去乙组？

(7.1)

##### 输入

无

#### 输出

甲组抽调去乙组的人数

### 2. 开学大采购

#### 题目描述

新学期开始了，学校计划采购一批新的篮球和排球用来上体育课。学校共有n元经费，咨询体育用品店得知篮球x元/个，排球y元/个，现要求篮球和排球都至少采购1个，n元经费全部用完，且篮球和排球的总数要超过50个，请问有哪些采购方案？（按照篮球从少到多，排球从多到少输出所有可行的方案）

#### 输入

三个整数，n、x、y用空格隔开分别代表总经费、篮球单价、排球单价

#### 输出

所有可行的采购方案，每个组方案有2个整数用空格隔开，第一个整数代表篮球的采购个数，第二个整数代表排球的采购个数

#### 样例输入

```
1000 25 15
```

#### 样例输出

```
1 65
4 60
7 55
10 50
13 45
16 40
19 35
22 30
```

### 3. 恐龙园买玩具

#### 题目描述

小明暑假来到恐龙园游玩，在恐龙园的礼物店里，有一些形形色色的小恐龙玩偶，小明想购买其中霸王龙和三角龙玩偶送给自己的5位好朋友。店员告诉小明，霸王龙玩偶只需要x元，三角龙玩偶只需要y元。小明有n元，希望两种恐龙都能购买，购买的霸王龙的数量 $\geq$ 三角龙的数量，购买的总数要在5个或者5个以上（这样才够分），而且不能有钱剩下。

请你编程帮助小明输出所有可能的购买方案，每组方案占1行，先输出霸王龙的数量，再输出三角龙的数量（霸王龙的数量从少到多，三角龙的数量从多到少）

#### 输入

三个整数n x y，分别代表总金额、霸王龙的单价、三角龙的单价

#### 输出

所有满足条件的购买方案，每组购买方案占1行，用空格隔开2个数分别代表霸王龙的数量和三角龙的数量。

#### 样例输入

```
100 10 5
```

## 样例输出

```
7 6  
8 4  
9 2
```

### 4. 买糕点

#### 题目描述

妈妈给了小明n元，给小明同学去面包店买糕点吃，小明非常喜欢吃切片面包和蛋挞，今天切片面包x元一件（一包），蛋挞y元一件（一盒）；小明想花光n元买这两样糕点，而且每样都至少买一件，请问，小明可以采购的方案中，能够买最多面包的方案是什么？

比如，100元，面包15元/件，蛋挞10元/件，那么可行的购买方案有：

2件面包 7件蛋挞

4件面包 4件蛋挞

6件面包 1件蛋挞

而上述方案中，面包最多的购买方案是：6件面包 1件蛋挞，因此输出：6 1

#### 输入

三个变量：n x y，分别代表总金额，面包的单价和蛋挞的单价！

#### 输出

两个数，分别代表采购方案中能够买到最多面包的件数和蛋挞的件数！

#### 样例输入

```
100 15 10
```

## 样例输出

```
6 1
```

## 4. 嵌套循环穷举

### 1. 百钱百鸡问题

#### 题目描述

用100元钱买100只鸡，公鸡，母鸡，小鸡都要有。公鸡5元1只，母鸡3元1只，小鸡1元3只。请问公鸡，母鸡，小鸡各应该买多少只？

#### 输入

无

#### 输出

每种买法占一行，由3个数组成，顺序为 公鸡数 母鸡数 小鸡数。每个数字空格隔开。

#### 示例代码

```
#include<bits/stdc++.h>  
using namespace std;  
int main(){  
    int i,j,x,y;  
    for(i=1;i<=(100-3-1)/5;i++){  
        x=100-i*5;  
        for(j=1;j<=(x-1)/3;j++){  
            y=x-j*3;
```

```

y=x-j*3;
if(i+j+y*3==100){
    cout<<i<<" "<<j<<" "<<y*3<<endl;
}
}
return 0;
}

```

## 5.课堂范例

### 1. 兑换硬币

#### 题目描述

用一张一元票换1分、2分和5分的硬币，每种至少一枚，问有几种换法？

#### 输入

无

#### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

#### 示例代码：

```

#include<iostream>
using namespace std;
int main(){
/*
一张一元票换1分、2分、5分的硬币。
每种至少一枚，问有几种换法？
1元=10角=100分
循环1分可能的数量范围，用剩余的钱换2分（循环2分的范围）
*/
int x , y , i , j , c = 0;
for(i = 1;i <= 100 - 2 - 5;i ++){
    x = 100 - i ;
    for(j = 1;j <= (x - 5) / 2;j ++){
        y = x - j * 2;
        if(y % 5 == 0){
            // cout<<i<<" "<<j<<" "<<y/5<<endl;
            c++;
        }
    }
}
cout << c;
return 0;
}

```

### 2. 购买文具

#### 题目描述

新学年就要开始了，爸爸把N元钱给了小青，让他购买一批文具，并作了以下要求：只能买圆珠笔、铅笔和铅笔芯，并且每样至少买一支，总数要超过30支，而且钱要全部花完。  
当小青去到文具店时，发现圆珠笔8角钱一支、铅笔2角钱一支、铅笔芯1角钱一支。小青怎么买才能符合爸爸的要求呢？请你编个程序帮他算出符合购买要求的所有方案总数。

### 输入

一个整数N，表示购买文具一共的元数。 (1 <= N <= 50)

### 输出

一个整数，即符合购买要求的所有方案总数。

### 样例输入

```
8
```

### 样例输出

```
135
```

```
/*
思路：
N元钱，每样至少买一支，总数要超过30支且钱要全部花完，
圆珠笔8角一支，铅笔2角一支，铅笔芯1角一支
*/
#include<iostream>
using namespace std;
int main(){
    //i表示圆珠笔个数
    //j表示铅笔个数
    //N表示输入的钱
    //c计数
    //x表示买完圆珠笔剩的钱
    //y表示买完铅笔剩的钱
    int N,x,y,i,j,c=0;
    cin>>N;
    N*=10;
    for(i=1;i<=(N-2-1)/8;i++){
        x=N-i*8;
        for(j=1;j<=(x-1)/2;j++){
            y=x-j*2;
            if(y+j+i>30){
                c++;
            }
        }
    }
    cout<<c;
    return 0;
}
```

## 6.课后作业

### 1. 搬砖问题

#### 题目描述

36块砖，36人搬。男搬4，女搬3，两个小儿抬一砖。要求一次全搬完。问需男、女、小儿各若干？

注意：假设男、女、小孩都有，请按照男、女、小孩的顺序输出所有可能的人数分配，每种人数分配方案占1行，每个数字空格隔开。

#### 输入

无

#### 输出

所有可能的人数分配方案，按照由小到大输出

### 2. 马克思手稿的问题

#### 题目描述

马克思手稿中有一道趣味数学题：有30个人，其中可能有男人、女人和小孩，在一家饭馆里吃饭共花了50先令，假设每个男人各花3先令，每个女人各花2先令，每个小孩各花1先令，问男人、女人和小孩各有几人？（注意：不一定男人、女人、小孩都有）

#### 输入

无

#### 输出

每行3个数，按照男人、女人、小孩的顺序，由小到大依次输出所有可能的人数方案（男人、女人、小孩其中某些人的数量可以为0）

### 3. 桐桐的计算

#### 题目描述

这个周末数学老师布置了一道有趣的题目，意思是：九头鸟（传说中的一种怪鸟，它有九个头，两只脚）、鸡和兔子关在一个笼子里。数数它们的头正好是100个，数数它们的脚也正好是100只。老师让桐桐编程计算其中九头鸟、鸡和兔子各有多少只，你能帮助桐桐吗？

#### 输入

无

#### 输出

前面若干行，每行输出满足题目条件的一个解，共三个数，分别表示九头鸟、鸡和兔子的只数，最后一行输出题目解的总数。

### 4. 怎样种树？

#### 题目描述

公园准备在小山上种桃树、梨树、苹果树，为了美观，总共准备种n棵树（ $n \geq 6$ 且n一定是6的倍数），要求三种树都得有，且每种树的数量都得是偶数，桃树的数量不能比梨树的数量多，梨树的数量不能比苹果树的数量多。请问有这三种树的数量分别有哪些可能的组合方法，从少到多分别数出桃树、梨树、苹果树可能的数量组合，每行1个方案。（6.1.99）

#### 输入

一个整数n（ $n \geq 6$ 且是6的倍数）

#### 输出

若干行的可能的组合方案，每行3个数，分别代表桃树、梨树、苹果树的可能的方案。

#### 样例输入

```
18
```

#### 样例输出

```
2 2 14
2 4 12
2 6 10
2 8 8
4 4 10
4 6 8
6 6 6
```

### 5. 桐桐去购物

#### 题目描述

桐桐周末陪妈妈到市场购物。她和妈妈来到一个买鸡的摊位，发现鸡的价格有三种：公鸡每只5元钱，母鸡每只3元钱，小鸡3只1元钱。妈妈就给桐桐出了一道计算题：如果用n元钱买m只鸡，问公鸡、母鸡和小鸡可以各买多少只？注意：必须把n元钱正好用完，且买的各种鸡的只数为大于等于0的整数。桐桐回到家里便拿起笔来认真计算，算了好久还没得出答案。聪明的你通过编写程序帮助桐桐找出结果好吗？

#### 输入

只有1行，两个数n和m ( $0 < n, m \leq 20000$ )。

#### 输出

有若干行，每行三个数，分别为公鸡、母鸡和小鸡的只数，用空格隔开，按照公鸡只数升序排列。

#### 样例输入

```
100 100
```

#### 样例输出

```
0 25 75
4 18 78
8 11 81
12 4 84
```

## 7. 规律类穷举

### 1. 四个人的年龄求解

#### 题目描述

张三、李四、王五、刘六他们四人的年龄是一个等差数列，且年龄相加是26，相乘是880，请问这四个人可能的年龄分别是多少？（假设一个人的年龄范围在1~130之间）

#### 输入

无

#### 输出

按照由小到大输出四个人的年龄的可能的值，数与数用空格隔开，每个可能的年龄方案一行，请输出所有可能的年龄方案！

## 2. 姐妹数对

### 题目描述

给定两个不同的正整数 $x, y$ ，若 $x+y$ 能被3除尽或能被7除尽，则称 $x, y$ 为姐妹数对。例如：

2,4; 2,5; 为姐妹数对。

3,14; 不是姐妹数对。

那么，对给出的一个整数 $n$ ( $1 \leq n \leq 100$ )， $1,2, \dots, n$ 之间有多少个姐妹数。

### 输入

一个整数 $n$

### 输出

一个整数，即 $1 \sim n$ 之间姐妹数对的个数。

### 样例输入

```
6
```

### 样例输出

```
8
```

## 3. 纸盒的最大体积是多少？

### 题目描述

在一张尺寸为 $n * n$ 厘米的正方形硬纸板的四个角上，分别裁剪掉一个 $m * m$ 厘米的小正方形，就可以做成一个无盖纸盒，请问这个无盖纸盒的最大体积是多少？

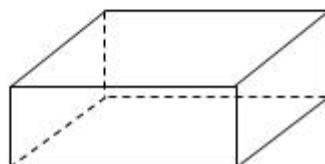
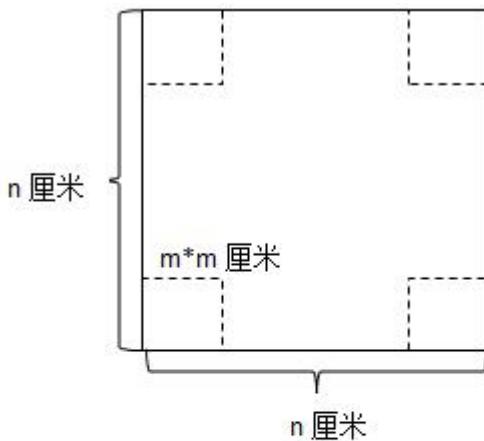
(立方体的体积 $V = \text{底面积} * \text{高}$ )

比如： $n = 5$ ，那么裁掉的小正方形的尺寸可能是1厘米、2厘米

如果裁掉1厘米的四个小正方形，得到纸盒的体积 =  $(5 - 2) * (5 - 2) * 1 = 9$ 立方厘米

如果裁掉2厘米的四个小正方形，得到纸盒的体积 =  $(5 - 4) * (5 - 4) * 2 = 2$ 立方厘米

因此，裁掉边长为2的四个小正方形得到的纸盒体积最大，最大体积为9 (立方厘米)



### 输入

一个整数n，代表正方形纸板的边长

### 输出

纸盒的最大体积

### 样例输入

5

### 样例输出

9

## 8.拓展作业

### 1. 回文数

#### 题目描述

回文数的定义为：如果把一个数的各个数位上的数字颠倒过来得到的新数与原数相等，则此数是回文数，例：7,22,131,2112,31013,...都是回文数。对任意给出的一个整数n,经过一系列的处理，最后都能成为回文数。处理的方法是，该数加上它的颠倒数，

例如：n=176

第一次处理后  $176+671 = 847$

第二次处理后  $847+748 = 1595$

第三次处理后  $1595+5951 = 7546$

第四次处理后  $7546+6457 = 14003$

第五次处理后  $14003+30041 = 44044$

此时成为回文数，共进行5次处理。

问题：给出n后，求出使该数按照以上规则进行一系列处理后成为回文数的最少操作次数。

### 输入

n 一个整数 ( $1 \leq n \leq 1000000$ )

### 输出

使n成为回文数的最少处理次数。若开始给出的n是回文数，则输出0（即不需任何处理）。

#### 样例输入

```
67
```

#### 样例输出

```
2
```

### 2. 一个六位数

#### 题目描述

有一个六位数，其个位数字7，现将个位数字移至首位（十万位），而其余各位数字顺序不变，均后退一位，得到一个新的六位数，假如新数为旧数的4倍，求原来的六位数。

#### 输入

无

#### 输出

原来的六位数

### 3. 猴子吃桃子

#### 题目描述

猴子吃桃子问题：猴子第一天摘下若干个桃子，当即吃了一半还不过瘾，又多吃了1个；第二天又将剩下的桃子吃掉一半又多吃了1个；以后每天早上都吃了前一天剩下的一半零1个。到了第十天想再吃时，见只剩下1个桃子，求第一天共摘了多少个桃子？

#### 输入

无

#### 输出

一个整数，第一天共有多少个桃子

## 第十三章 数组

### 1. 什么是数组？

#### 入门问题：

考试结束，尹老师想统计一下同学们中成绩优秀 ( $>=90$ )的人和不及格 ( $<60$ )的人，已知班上有20人。请编程实现读入20人成绩并帮助尹老师完成统计。

#### 难点分析：

有20人需要计算成绩，按原先的思路需要定义20个变量，但是如果问题变成100人，1000人该如何处理？

#### 解决方法：

定义一个长度为20的**数组**，存放这20位同学的成绩，即可实现一个**变量存储多个元素的目的**。

**数组：连续存储的相同数据类型元素的集合。**

#### 注意：

普通变量每次只能存储一个数据，比如 `int x = 10`; `x` 只能存储一个整数，而如果需要存储大量的相同类型数值的时候(比如上面的问题)，就需要使用数组

### 例子：数组的定义、赋值和输出

```
#include<iostream>
using namespace std;
int main(){
    int x = 10;
    //定义一个长度为5的整数数组，存放5个人的成绩
    //定义变量时，[5]代表数组长度是5
    //定义数组是，如果不为数组赋值，那么数组元素的值是无法确定的，
    //原理同第二章第2节第3点一样
    int a[5];
    a[0] = 100;
    a[1] = 98;
    a[2] = 95;
    a[3] = 80;
    a[4] = 92;
    //输出数组中每个元素的值
    //访问数组时，[0]代表数组下标
    cout << a[0] << endl;
    cout << a[1] << endl;
    cout << a[2] << endl;
    cout << a[3] << endl;
    cout << a[4] << endl;
    /*
    遍历数组的方式：
    for(int i=0;i<5;i++){
        cout<<a[i]<<endl;
    }
    */
    return 0;
}
```

**注意：**长度为n的数组，下标从n开始，到n-1结束，不能超过n-1，如果超出范围则会产生“数组下脚标越界”的语法错误。这可能（不是一定）导致程序崩溃。

例题：定义长度为5的数组，存储一个班同学的成绩(数据自行决定)，问该班同学的总分和平均分分别是多少？

**注意：**如果定义时只给数组第一个元素赋值（或者说没有全部赋值但至少一个）的话，未赋值的元素会默认为0。

**案例：从键盘读入数组的元素：**

### 考试成绩的简单统计

#### 题目描述

期末考试结束，王老师想知道这次考试中成绩优秀的学生有多少人（考试成绩大于或等于90表示成绩优秀），请你编程帮助王老师来计算出成绩优秀的人数。

#### 输入

第一行，一个整数n代表有n个人的成绩 (n<=100)

第二行，n个人的成绩，用空格隔开

## 输出

成绩优秀的同学的总人数

## 样例输入

```
5
98 88 85 99 90
```

## 样例输出

```
3
```

a[i]的值 数组元素的值	98	88	85	99	90	...
下角标，即i的值	0	1	2	3	4	...

**写法一：**一个循环做循环读入数组元素的值，再做一个循环作为分数统计

```
#include<iostream>
using namespace std;
int main(){
    int a[100] = {0};
    int n , i , c = 0;
    for(i = 0;i < n;i ++){
        cin >> a[i];
    }
    for(i = 0;i < n;i ++){
        if(a[i] >= 90){
            c++;
        }
    }
    cout << c << endl;
    return 0;
}
```

**写法二：**每读入一个数据就判断是否是满足条件：

```
#include<iostream>
using namespace std;
int main(){
    int a[100] = {0};
    int n , i , c = 0;
    for(i = 0;i < n;i ++){
        cin >> a[i];
        if(a[i] >= 90){
            c++;
        }
    }
    cout << c << endl;
    return 0;
}
```

```
}
```

## 2.课堂练习

### 1. 查找“支撑数”

#### 题目描述

在已知一组整数中，有这样一种数非常怪，它们不在第一个，也不在最后一个，而且刚好都比左边和右边相邻的数大，你能找到它们吗？

#### 输入

第一行为整数m，表示输入的整数个数。 ( 3<= m <=100 ) 第二行为m个整数。

#### 输出

若干个支撑数，每行一个。

#### 样例输入

```
14
1 3 2 4 1 5 3 9 7 10 8 23 85 43
```

#### 样例输出

```
3
4
5
9
10
85
```

**思路：**由于第一个和最后一个都无法成为支撑数，所以我们只需循环 1 ~ n - 2 的下脚标。对于下脚标为i的元素a[i]而言，左侧的数位a[i-1]，右侧的数字为a[i+1]。

#### 示例代码：

```
#include<iostream>
using namespace std;
int main(){
    /*
    不在第一个也不在最后一个,
    且比相邻两个数字都大
    */
    int a[100] , n , i ;
    cin >> n;
    //输入数据时候不能去掉两头
    for(i = 0;i < n ;i ++){
        cin >> a[i];
    }
    //寻找支撑数
    for(i = 1;i < n-1;i ++){
        if(a[i] > a[i-1] && a[i] < a[i+1]){
            cout << a[i];
        }
    }
    return 0;
}
```

```
}
```

**补充：长度为n的数组，下标为0 ~ n-1，如果在程序中访问a[i-1]就不可以才能够0开始，同理访问a[i+1]就不可以循环到n-1**

## 2. 排除异形基因

### 题目描述

神舟号飞船在完成宇宙探险任务回到地球后，宇航员张三感觉身体不太舒服，去了医院检查，医生诊断结果：张三体内基因已被改变，原有人体基因序列中已经被渗入外星球不明异形生物基因，但可喜的是，这些异形基因都有一个共同的特征，就是该基因序号的平方除以7的余数都是1，要赶快清除掉，否则会危害整个人类。赶快行动吧。

### 输入

第一行是一个整数n（基因个数,3<=n<=200） 第二行是n个整数（张三的基因序列）

### 输出

去除异形基因后的正常序列，空格隔开

### 样例输入

```
4
6 2 8 12
```

### 样例输出

```
2 12
```

```
#include<iostream>
using namespace std;
int main(){
    /*
    思路：
    循环数组的每个元素，如果满足条件就是好基因，则输出
    */
    int a[1000],n,i;
    cin >> n;
    for(i = 0;i < n;i ++){
        cin >> a[i];
    }
    for(i = 0;i < n;i ++){
        if(a[i] * a[i] % 7 != 1){
            cout << a[i] << " ";
        }
    }
    return 0;
}
```

## 3. 找找谁的身高超过全家的平均身高

### 题目描述

找找谁的身高超过全家的平均身高。全家n口人，输入输出数据如下：(平均身高保留一位小数)

### 输入

第一行有一个整数n( 1 < n < 11 )。第二行是n个整数，用空格隔开。

### 输出

第一行为全家的平均身高 (保留一位小数) ； 第二行有若干个数，为超过平均身高的的人的身高厘米数。

### 样例输入

```
7  
175 160 172 158 178 162 142
```

### 样例输出

```
AVE=163.9  
1:175 3:172 5:178
```

### 示例代码：

```
#include<iostream>  
#include<iomanip>  
using namespace std;  
int main(){  
    /*  
    思路：  
    第一步：求和；  
    第二步：求平均；  
    第三步：求哪些人在平均值以上  
    */  
    int n , a[10] , i , s = 0;  
    cin >> n;  
    for(i = 0;i < n;i ++){  
        cin >> a[i];  
        s += a[i];  
    }  
    double ave = s * 1.0 / n ;  
    cout << fixed << setprecision(2) << "AVE=" << ave << endl;  
    for(i = 0;i < n;i ++){  
        if(a[i] > ave){  
            cout << i + 1 << ":" << a[i] << " ";  
        }  
    }  
    return 0;  
}
```

## 3.课后作业

### 1. 考试成绩的分布情况

#### 题目描述

期末考试结束，小明的语文老师想知道，这次考试的成绩分布情况，主要计算如下几个数据：平均分、 $\geq$ 平均分的总人数、 $<$ 平均分的总人数，请你写程序帮助小明的语文老师来计算一下！

#### 输入

第一行，一个整数n代表有n个人的成绩 (n<=100)

第二行，n个人的语文成绩

#### 输出

3个值，分别代表平均分、 $\geq$ 平均分的总人数、 $<$ 平均分的总人数，请注意，平均分保留1位小数！

#### 样例输入

```
5  
100 98 97 99 90
```

#### 样例输出

```
96.8 4 1
```

### 2. 打折优惠

#### 题目描述

商场周末大优惠，规定凡购物超过100元时，超过100元那部分便可打9折。小雄同妈妈一起购买了一大批物品，你能帮他算出最终的应付款吗？

#### 输入

第一行一个整数N，表示所买物品的个数。第二行N个空格隔开的整数，表示N件物品要付的元数。

#### 输出

最终的应付款。（保留两位小数）

#### 样例输入

```
5  
60 80 50 30 20
```

#### 样例输出

```
226.00
```

### 3. 橘子称重

#### 题目描述

学校买回来一大箱橘子，有m个 ( $m \geq 100 \&& m \leq 1000$ )，橘子大小比较均匀，学校想称一下总共有多重，发现大称坏掉了还没有修好，只有一个小小的弹簧秤。学校又不想分开称，那样太慢了。

小明想了一个办法，由于橘子大小比较均匀，可以从中拿n个出来 ( $n \geq 5 \&& n \leq 20$ )，这n个橘子的重量弹簧秤是可以称出来的，有了这n个橘子的重量，就可以计算出平均一个橘子有多重，这样就能知道整箱大约有多重了。

请编写程序，从键盘读入橘子总数m，小明称的橘子的个数n以及这n个橘子的重量，计算出这箱橘子总共约有多重（结果保留1位小数）。(4.1.52)

#### 输入

2行，第一行2个整数m和n，分别代表一箱橘子的总个数以及小明称的橘子的个数。

第二行为n个橘子的重量（整数）。

#### 输出

一箱橘子的重量（保留1位小数）

## 样例输入

```
100 7  
84 83 82 81 80 79 79
```

## 样例输出

```
8114.3
```

### 4. 陶陶摘苹果

#### 题目描述

陶陶家的院子里有一棵苹果树，每到秋天树上就会结出10个苹果。苹果成熟的时候，陶陶就会跑去摘苹果。陶陶有个30厘米高的板凳，当她不能直接用手摘到苹果的时候，就会踩到板凳上再试试。

现已知10个苹果到地面的高度，以及陶陶把手伸直的时候能够达到的最大高度，请帮陶陶算一下她能够摘到的苹果的数目。假设她碰到苹果，苹果就会掉下来。

#### 输入

包括两行数据。第一行包含10个100到200之间（包括100和200）的整数（以厘米为单位）分别表示10个苹果到地面的高度，两个相邻的整数之间用一个空格隔开。第二行只包括一个100到120之间（包含100和120）的整数（以厘米为单位），表示陶陶把手伸直的时候能够达到的最大高度。

#### 输出

包括一行，这一行只包含一个整数，表示陶陶能够摘到的苹果的数目。

## 样例输入

```
100 200 150 140 129 134 167 198 200 111  
110
```

## 样例输出

```
5
```

### 5. 求和

#### 题目描述

输入n ( $1 \leq n \leq 5000$ ) 个正整数，每个数都在1到20000之间；要求对这n个数中的奇数和偶数分别求和。

#### 输入

第一行，一个正整数n ( $1 \leq n \leq 5000$ )；第2-n+1行，每行一个正整数，每个数都在1到20000之间。

#### 输出

输出文件共有二行，每行包含一个整数，第一行为所有奇数之和，第二行为所有偶数之和。

## 样例输入

```
5  
3  
10  
7  
5  
8
```

### 样例输出

```
15  
18
```

## 6. 完美的偶数

### 题目描述

完美偶数指的是，如果一个数本身是偶数，且这个数是偶数位的数，且这个数的各个位也是偶数，那么这个数就可以称为完美偶数；比如：28就是完美偶数，而246就不是，因为246是一个3位数。请你编程求出，从键盘读入的n个数中，哪些数是完美的偶数，输出他们。

### 输入

第一行是一个整数n ( $n \leq 100$ )  
第二行是n个整数（这些整数都是1~9999范围内的整数）

### 输出

按顺序输出这n个数中的完美偶数，每个数一行

### 样例输入

```
5  
26 4286 228 32 1280
```

### 样例输出

```
26  
4286
```

## 7. 输出奇数和偶数

### 题目描述

输入n个整数，将其中的奇数和偶数分别显示出来 ( $1 < n < 30$ )

### 输入

第一行：一个整数n。第二行：n个空格隔开的整数。

### 输出

第一行：若干个奇数。第二行：若干个偶数。（每个数前面都有一个空格）

### 样例输入

```
10  
21 12 33 43 59 68 77 18 19 40
```

### 样例输出

```
21 33 43 59 77 19  
12 68 18 40
```

## 8. 拿到某个数的概率是多少?

### 题目描述

老师在一个不透明的纸袋里放入一些乒乓球，每个乒乓球上都有一个数字，当球放入之后，让小明从中随机拿一个。在球放入之后，小明抽之前，老师想让您帮忙编程先计算一下，拿到某个数字x的概率是多少？

比如：老师向袋子里面放入了5个球，对应的数字分别是2 3 2 4 2，那么拿到数字2的概率为 $3 / 5 = 0.6$ ，拿到数字3的概率为 $1 / 5 = 0.2$ 。（8.2）

### 输入

第一行是一个整数n代表袋子里球的个数

第二行n个数，代表袋子里每个球上面的数字的值

第三行一个整数x，是要计算出现概率的数字（x不一定在第二行的n个数中出现，也就是说x对应的球可能在袋子里没有）

### 输出

x被拿到的概率（结果保留2位小数）

### 样例输入

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

### 样例输出

```
0.60
```

### 提示

拿到数字x的概率 = 数字x的个数 / 总数字的个数

第一步：求出数字x在数组中出现的次数

第二步：概率 = x出现的次数 / 总元素个数

```
#include <bits/stdc++.h>  
using namespace std;  
  
int main() {  
    int a[1000]; //存储读入的数字  
    int i, n, c = 0, x; //c统计x在数组中出现的次数  
    cin >> n; //读入的数字的数量  
    //读入n个数  
    for(i = 0; i < n; i++) {  
        cin >> a[i];  
    }  
    //读入x, 表示要找的数  
    cin >> x;  
  
    //求出x在数组中出现了几次  
    for(i = 0; i < n; i++) {
```

```
if(a[i] == x){  
    c++;  
}  
  
//输出x出现的概率  
cout<<fixed<<setprecision(2)<<c * 1.0 / n;  
return 0;  
}
```

## 9. 哪个厂家的零件更标准?

### 题目描述

在统计描述中，方差用来计算每一个变量（观察值）与总体均数之间的差异。比如：甲乙2个厂商生产某零件，一批零件要求在尺寸合格的情况下，大小越一致越好，由于生产工艺的问题，零件生产厂商生产的零件不可能一模一样。

为了检测甲乙两个厂商，那个厂商生产的零件更符合标准，分别从2个厂商生产的零件中抽取5个样品尺寸如下：

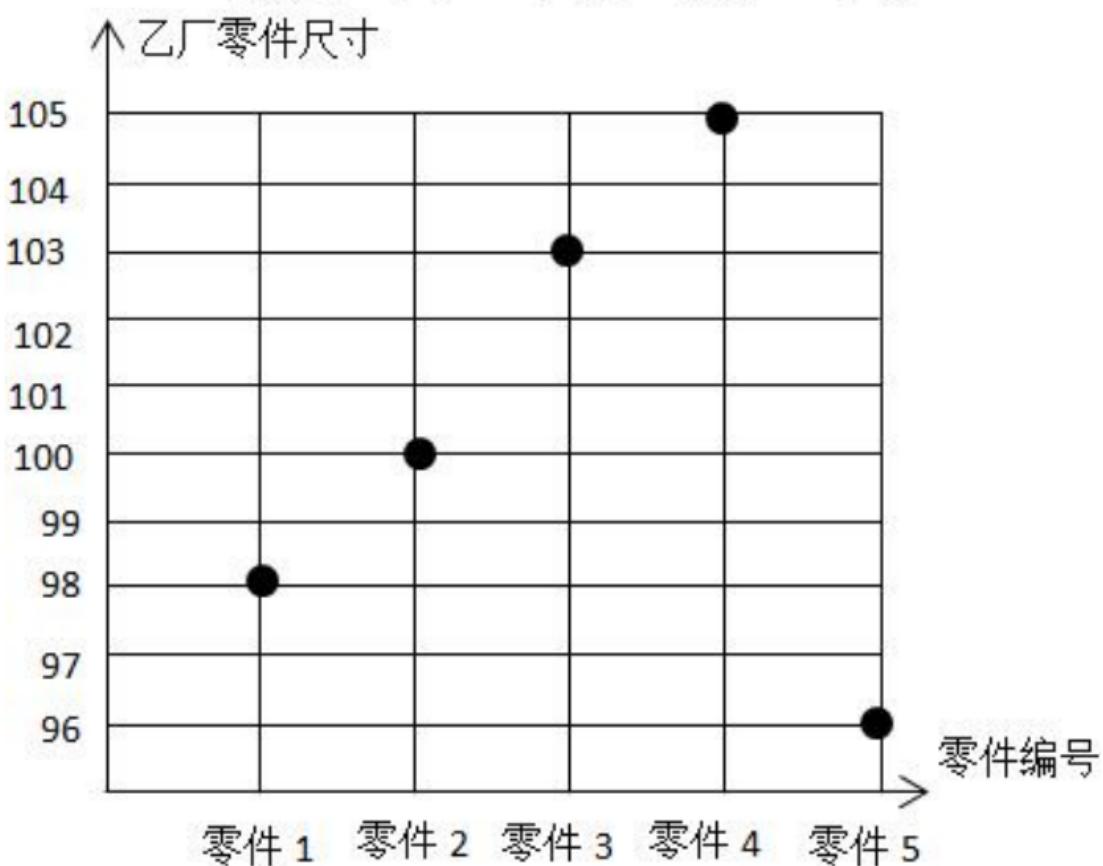
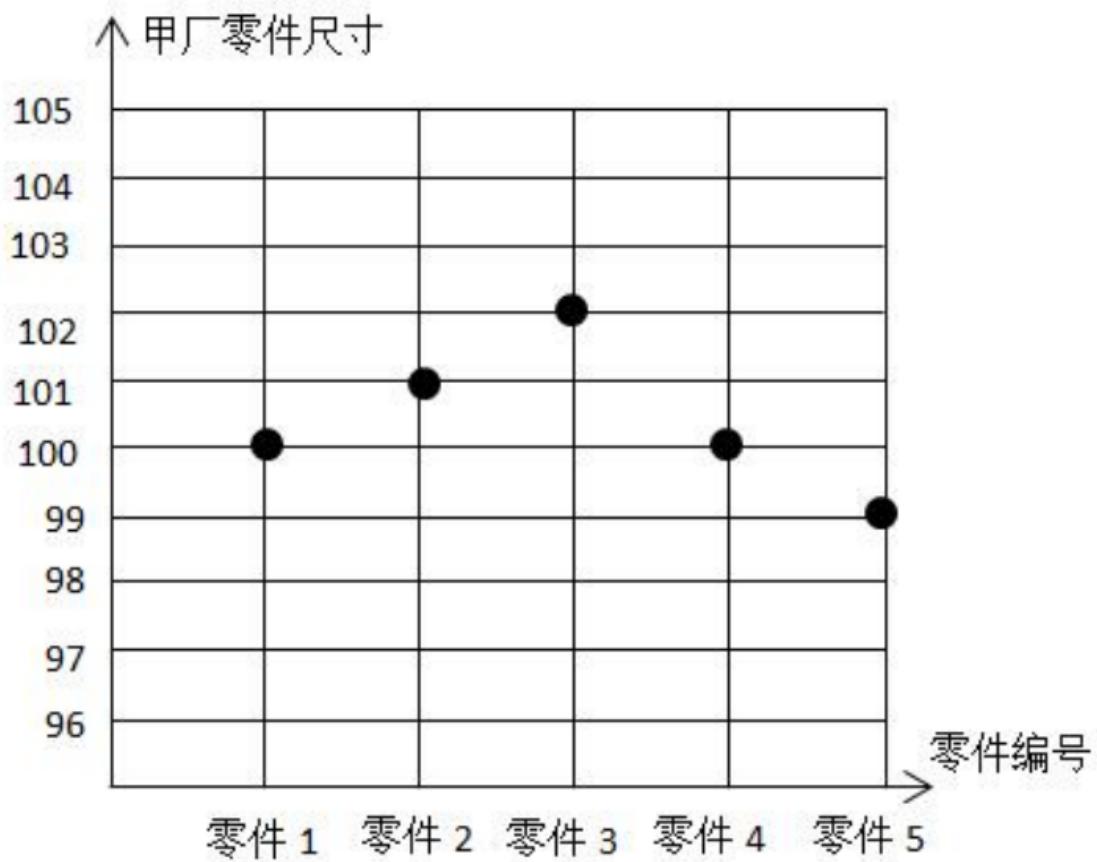
甲：100 101 102 100 99

乙：98 100 105 103 96

假设零件尺寸在95~110之间都算合格，那么两批零件都是合格的。

如果按照平均数计算，两组数据的平均值都是100.4

但如果将两组数据画到数轴上：



从两个厂抽检的零件分布图可以看出，甲厂的零件大小更加一致，更加符合标准。

为了方便计算数据的离散程度，我们引入方差的概念，方差的计算公式为：

$$s^2 = [(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \dots + (x_n - \bar{x})^2]$$

其中 $x_1 \sim x_n$ 代表一组数据中的每个元素， $\bar{x}$ 代表这组数据的平均值。

按照公式，甲厂零件的方差为：

$(100-100.4)^2 + (101-100.4)^2 + (102-100.4)^2 + (100-100.4)^2 + (99-100.4)^2 = 5.2$

乙厂零件的方差为：

$(98-100.4)^2 + (100-100.4)^2 + (105-100.4)^2 + (103-100.4)^2 + (96-100.4)^2 = 53.2$

从方差上也可以看出，甲厂的零件更符合标准！

现从键盘读入2个厂生产的零件尺寸（假设零件的尺寸都是合格的），请计算哪个厂的零件尺寸更加一致（方差更小）？

### 输入

第一行为一个整数n，代表2个厂抽检的零件的个数！（n在5~100之间）

第二行为甲厂的n个零件的尺寸

第三行为乙厂的n个零件的尺寸

### 输出

哪个厂的零件更加符合标准，甲厂请输出“jia”，乙厂请输出“yi”

### 样例输入

```
5
100 101 102 100 99
98 100 105 103 96
```

### 样例输出

```
jia
```

## 4. 找数问题

### 1. 数组元素的查找

#### 题目描述

给你m个整数，查找其中有无值为n的数，有则输出该数第一次出现的位置，没有则输出-1。

#### 输入

第一行一个整数m：数的个数（ $0 \leq m \leq 100$ ）第二行m个整数（空格隔开）（这些数在0-999999范围内）第三行为要查找的数n

#### 输出

n的位置或-1

### 样例输入

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

### 样例输出

```
3
```

#### 思路：

循环数组的每个元素，判断是否是要找的元素n，如果是则输出下标i+1，如果找不到就输出-1。

### 解法一：通过变量f来标记是否曾经找到过n (结束后才可以得知结果)

```
#include<iostream>
using namespace std;
int main(){
    int m , n , i , a[100];
    cin >> m ;
    for(i = 0;i < m;i ++){
        cin >> a[i];
    }
    cin >> n;
    int f = 0;//f=0表示没找到;
    for(i = 0;i < m;i ++){
        if(a[i] == n){
            f = 1 ;
            cout << i + 1;
            break ;
        }
    }
    if(f == 0){
        cout << -1;
    }
    return 0;
}
```

### 解法二：使用变量p来存储n第一次出现的位置，设置p的初始值为-1 (最终如果值为-1表示没找到)

```
#include<iostream>
using namespace std;
int main(){
    int m , n , i , a[100] , p = -1;
    cin >> m;
    for(i = 0;i < m;i ++){
        cin >> a[i];
    }
    cin >> n;
    for(i = 0;i < m;i ++){
        if(a[i] == n){
            p = i + 1;
            break; //一定要加break!
        }
    }
    cout << p;
    return 0;
}
```

## 2. 求n个数的最大值和最小值

### 题目描述

任意输入n个整数，把它们的最大值，最小值求出来。

### 输入

输入只有一行，包括一个整数n(1<=n<=20),后面跟着n个数。每个数的范围在0到32767之间。

### 输出

输出只有一行,包括2个整数。

### 样例输入

```
5 1 99 3 6 0
```

### 样例输出

```
99 0
```

### 思路1:

假设第一个数字是最大的, 循环后面所有的元素, 如果有比最大的数字还大的, 就替换最大数。  
(打擂法)

max:存放最大数;

min:存放最小数;

max初始值假设为 a[0]

### 示例代码:

```
#include<iostream>
using namespace std;
int main(){
    int a[20], n, i, max, min;
    cin >> n;
    for(i = 0; i < n; i++){
        cin >> a[i];
    }
    max = a[0];
    min = a[0];
    //循环比较这些元素, 如果有比max大的就换掉max
    //需要注意的是循环无需从a[0]开始, 因为a[0]已经假设为最大/小了
    for(i = 1; i < n; i++){
        if(a[i] > max){
            max = a[i];
        }
        if(a[i] < min){
            min = a[i];
        }
    }
    cout << max << " " << min;
    return 0;
}
```

**注意:** 如果拿到了最大值, 是无法立马得到最大值的下脚标的, 而如果得到了最大值的下脚标则可以一下子拿到最大值。因此我们求最大值的时候选择求最大值的下脚标更方便。

### 思路2:

如果要找到最大数的下脚标, 需要假设max = 0时对应的值最大, 即第一个数据是最大的, 循环比  
较后续所有元素, 如果  $a[i] > a[max]$ , 将 max设为i。

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

```
int main(){
    int a[20], n, i, max = 0, min = 0;
    cin >> n;
    for(i = 0; i < n; i++){
        cin >> a[i];
    }
    for(i = 1; i < n; i++){
        if(a[i] > a[max]){
            max = i;
        }
        if(a[i] < a[min]){
            min = i;
        }
    }
    cout << a[max] << " " << a[min];
    return 0;
}
```

## 5.课后作业

### 1. 歌唱比赛评分

#### 题目描述

四（1）班要举行一次歌唱比赛，以选拔更好的苗子参加校的歌唱比赛。评分办法如下：设N个评委，打N个分数（ $0 \leq$  每个分数  $\leq 10$ ），去掉一个最高分，去掉一个最低分，剩下的评委的平均分即为该选手的最后得分。但是选手太多了，靠人工计算每个选手的得分太慢太麻烦。你能不能帮帮他们，设计一个程序让计算机来算出选手的最后得分呢？（4.1.53）

#### 输入

第一行为一个整数N ( $5 \leq N \leq 10$ )  
第二行为N个整数Ai ( $0 \leq Ai \leq 10$ )

#### 输出

选手的最后得分（保留两位小数）

#### 样例输入

```
5
5 6 7 8 9
```

#### 样例输出

```
7.00
```

### 2. 最大数

#### 题目描述

n个数中最大的那个数在哪里？输出其位置，若有多个最大数则都要输出。

#### 输入

第一行：n ( $3 \leq n \leq 10$ ) 第二行：空格隔开的n个数

#### 输出

输出若干个数，表示最大数的位置，每行一个。

### 样例输入

```
5  
1 2 6 3 6
```

### 样例输出

```
3  
5
```

## 3. 摘苹果

### 题目描述

小红来到苹果园，帮园长摘苹果，园长请小红把摘完的苹果的最小的那个去掉（如果有多个最小的苹果，那么都要去掉），剩余的苹果算一下平均一个苹果有多重？（平均重量请保留1位小数）

### 输入

输入有2行：

第一行：一个整数n代表小红摘的n个苹果！

第二行：n个苹果的重量（分别用n个整数表示）！

### 输出

去掉最小的苹果后，摘到的苹果的平均重量！

### 样例输入

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

### 样例输出

```
2.7
```

## 6. 数组元素的移动

### 1. 数组逆序

### 题目描述

给你m个整数，将其逆序输出

### 输入

第一行一个整数m ( $3 \leq m \leq 100$ )：数的个数 第二行m个整数（空格隔开）（这些数在0-9999999之间）

### 输出

m个整数（空格隔开）

### 样例输入

```
3  
1 7 5
```

### 样例输出

5 7 1

### 思路1：

直接逆序输出

```
#include<iostream>
using namespace std;
int main(){
    int a[100],i,n;
    cin >> n;
    for(i = 0;i < n;i ++){
        cin >> a[i];
    }
    for(i = n - 1 ; i >= 0 ; i --){
        cout << a[i] << " ";
    }
    return 0;
}
```

### 思路2：

真逆序，取数组长度一半，对称位置互换。

10	20	30	40	50
0	1	2	3	4

m = 5

a[0]-->a[4]

a[1]-->a[3]

归纳可得：下脚标为i 的元素应该和下角标为  $n - i - 1$  的元素互换。

示例代码：

```
#include<iostream>
using namespace std;
int main(){
    int a[100],i,n,t;//t用作中转变量
    cin >> n;
    for(i = 0;i < n;i ++){
        cin >> a[i];
    }
    for(i = 0;i < n / 2;i ++){
        t = a[i];
        a[i] = a[n-i-1];
        a[n-i-1] = t;
    }
    for(i = 0 ; i < n ; i ++){
        cout << a[i] << " ";
    }
    return 0;
}
```

## 2. 数组元素的删除

### 题目描述

把一个数组的第x个位置的元素删除掉

### 输入

有三行

第一行有一个整数n( n <= 10 )

第二行有n个整数

第三行有一个整数x, 为要删除的位置

### 输出

输出更新后的数组

### 样例输入

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

### 样例输出

```
1 2 4 5
```

### 思路1:

不真删除, 遇到脚标为x的元素时不输出

```
#include<iostream>
using namespace std;
int main(){
    int a[100], i, n, x;
    cin >> n;
    for(i = 0; i < n; i++){
        cin >> a[i];
    }
    cin >> x;
    for(i = 0; i < n; i++){
        if(i+1 != x){
            cout << a[i] << " ";
        }
    }
    return 0;
}
```

### 思路2:

真删除, 即将x位置后面的所有数据向前移动一位.

a[3]=a[4];a[4]=a[5]... ...a[i]=a[i+1];

```
#include<iostream>
using namespace std;
int main(){
    int a[100], i, n, x;
    cin >> n;
```

```

for(i = 0;i < n;i ++){
    cin >> a[i];
}
cin >> x;
x -= 1;
for(i = x;i < n-1;i ++){
    a[i] = a[i+1];
}
for(i = 0;i < n - 1;i ++){
    cout << a[i] << " ";
}
return 0;
}

```

### 3. 数组元素的插入

#### 题目描述

在一个数组的第 $x$ 个位置插入一个新的数 $y$

#### 输入

有四行 第一行有一个整数 $n$  ( $5 \leq n \leq 10$ ) 第二行有 $n$ 个整数 第三行有一个整数 $x$ , 为要插入的位置 第四行有一个整数 $y$ , 为要插入的整数

#### 输出

更新后的数组

#### 样例输入

```

5
7 2 3 4 5
2
9

```

#### 样例输出

```

7 9 2 3 4 5

```

#### 思路:

逆序循环下脚标为  $n-1 \sim x$  的元素, 将这些元素后移, 空出的 $x$ 的位置插入元素 $y$ 即可。倒序规律为:  
 $a[i+1]=a[i]$

```

#include<iostream>
using namespace std;
int main(){
    int a[11],n,x,y,i;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    cin>>x>>y; //输入要插入的位置及具体的值
    x--;
    for(i=n-1;i>=x;i--){
        a[i+1]=a[i];
    }
    a[x]=y;
}

```

```
n++;  
for(i=0;i<n;i++){  
    cout<<a[i]<<" "  
}  
return 0;  
}
```

#### 4. 元素插入有序数组

##### 题目描述

给你一个整数n和一个数列(数列个数不超过1000), 这个数列保证从小到大排列, 现要求将这个整数n插入到数列中, 使新的数列仍然从小到大排列。

##### 输入

第一行一个整数n : 等待插入的数 ,第二行一个整数m : 数列中数的个数 ,第三行m个整数 (空格隔开)

##### 输出

一行整数: 新的数列 (空格隔开)

##### 样例输入

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

##### 样例输出

```
1 2 3 4 5
```

##### 思路:

1. 找出要插入元素的下标x, 从第一个数字开始逐个比较, 直到 $a[i] \geq n$ , 下标i就是x的值;
2. 将元素后移 (逆序循环  $n-1 \sim x$ )
3. 在x的位置插入元素n

##### 示例代码

```
#include<iostream>  
using namespace std;  
int main(){  
    int a[1001],n,m,i,x;  
    cin>>n>>m;  
    for(i=0;i<m;i++){  
        cin>>a[i];  
    }  
    //此举是为了防止下面的循环结束后没有值比n大,  
    //这时候n就要排在最后一个位置  
    x=m-1;  
    for(i=0;i<m;i++){  
        if(a[i]>=n){  
            x=i;  
            break;  
        }  
    }
```

```
for(i=m-1;i>x;i--) {
    a[i+1]=a[i];
}
a[x]=n;
m++;
for(i=0;i<m;i++){
    cout<<a[i]<<" ";
}
return 0;
```

## 7.课后作业

### 1. 删除数组的最小数

#### 题目描述

在一个不重复的数组中，请将这个数组的最小数删除后输出！

#### 输入

有两行 第一行有一个整数n (  $5 \leq n \leq 100$  ) 第二行有n个不重复的整数！

#### 输出

删除最小数后的数组！

#### 样例输入

```
5
1 7 6 8 2
```

#### 样例输出

```
7 6 8 2
```

### 2. 最小数

#### 题目描述

输入n个整数的数列，请找出数列中最小数所在的位置（若有多个最小数，则选最左边的那个最小数），把它与数列的第一个数对调，其他数的位置不动，输出此数列。

#### 输入

数组数的个数n (  $N \leq 200$  ) 一行n个数，用空格分开（都  $\leq 32767$  ）

#### 输出

第一行：最小数所在的位置（只需要输出最左边的一个的位置） 第二行：交换后的数组（一个空格隔开）

#### 样例输入

```
3
2 6 1
```

#### 样例输出

```
3  
1 6 2
```

### 3. 移动数组元素

#### 题目描述

在一个不重复的数组中，请将这个数组的最小数和数组第一个数交换，最大数和数组最后一个数交换！

#### 输入

有两行 第一行有一个整数n ( $5 \leq n \leq 100$ ) 第二行有n个不重复的整数！

#### 输出

移动位置后的数组！

#### 样例输入

```
5  
6 7 1 10 4
```

#### 样例输出

```
1 7 6 4 10
```

### 4. 在最大数后面插入一个数

#### 题目描述

在一个不重复数组的最大数的后面插入一个新的数y

#### 输入

有三行 第一行有一个整数n ( $5 \leq n \leq 100$ ) 第二行有n个整数 第三行有一个整数y，为要插入的数

#### 输出

更新后的数组

#### 样例输入

```
5  
7 2 3 4 5  
9
```

#### 样例输出

```
7 9 2 3 4 5
```

### 5. 小明排队做操迟到

#### 题目描述

做操的时间到了，小明在教室还在思考刚刚老师讲的一道题目，当他想通这个题时，同学们都已经在操场上排好队了，他赶快跑到操场上找到自己的班级队伍，希望尽快找到以前排队的位置，准备做操，小明记得应该排在第x学号同学的后面。你能不能来帮帮小明呢？

#### 输入

输入（两行）：

第一行3个整数：n x y (现在队伍的长度、第x同学的学号、小明的学号)

第二行n个整数：n个同学的学号

### 输出

n+1个学号 (小明加入队伍后队伍中的每个数据)

### 样例输入

```
4 32 23
1 8 32 56
```

### 样例输出

```
1 8 32 23 56
```

## 6. 换位置

### 题目描述

体育课上，有一个班级的同学站成了一队，体育老师请最高的和最矮的两位同学调换一下位置，其余的同学不要动，请编程实现！（假设所有人的高矮都是不一样的）

### 输入

第一行有一个整数n代表该班级的总人数

第二行有n个数，代表每个人的身高

### 输出

调换位置后的结果

### 样例输入

```
8
8 9 10 4 7 6 5 3
```

### 样例输出

```
8 9 3 4 7 6 5 10
```

## 第十四章 数组的进阶应用

### 1. 排序

#### 1. 数组元素的排序

### 题目描述

对数组的元素按从小到大进行排序

### 输入

有两行 第一行有一个整数n( 5 <= n <= 10 ) 第二行有n个整数

### 输出

输出更新后的数组

### 样例输入

8

1 2 3 6 8 7 4 5

## 样例输出

1 2 3 4 5 6 7 8

## 思路

1. 如果只需要替换两个元素的位置，我们需要引入一个中间变量 t ,  $t = a[0]$  ,  $a[0] = a[1]$  ,  $a[1] = t$ ;
2. 归纳可得 $a[n]$  与  $a[i]$  位置互换方式为：  $t = a[n]$  ,  $a[n] = a[i]$  ,  $a[i] = t$ ;
3. 按照上述方法，我们可以通过逐个比较的方式得到一个最大/最小值
4. 循环逐个比较，即可找出第二大/小，第三大/小，第四大/小...
5. 需要注意的是，我们每比较一轮，需要比较的次数就会少一次，因为在上一轮已经得到了一个上一轮中的最大/小值

### \*\*\*观看舞蹈 \*\*\*

<https://www.bilibili.com/video/BV1Px411L7Lu?from=search&seid=4193610121451017492>

## 示例代码

```
#include<iostream>
using namespace std;
int main(){
    //数组, t为中间变量, i、n为循环用变量
    int a[11],t,n,i,j;
    cin>>n;
    //给数组赋值
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    //外层循环控制比较多少轮，元素有n个，需要比较n-1轮，因为最后一个无需比较
    for(i=0;i<n-1;i++){
        //内循环控制每轮比较多少次，每比一轮，每一轮需要比较的次数就少一次
        for(j=0;j<n-i-1;j++){
            //如果满足置换条件，则执行置换
            if(a[j]>a[j+1]){
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
    //输出数组
    for(i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    return 0;
}
```

## 2. 数的排序

### 题目描述

输入n个不超过30000的整数 ( $n \leq 10$ )。然后求出每个数的数字和，再按每个数的数字和由小到大排列输出。

### 输入

第一行为整数n

第二行为n个整数

### 输出

由小到大排列的每个数的数字和 (每个数之间保留一个空格)

### 样例输入

```
4
33 104 87 16
```

### 样例输出

```
5 6 7 15
```

### 思路

1. 先将得到的数据更新为变换后的结果 (拆位计算)；
2. 对新的数据进行排序

```
#include<iostream>
using namespace std;
int main(){
    int a[11],t,n,i,j,g,s,b,q,w;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    //更新数据
    for(i=0;i<n;i++){
        g=a[i]%10;
        s=a[i]/10%10;
        b=a[i]/100%10;
        q=a[i]/1000%10;
        w=a[i]/10000;
        a[i]=g+s+b+q+w;
    }
    for(i=0;i<n-1;i++){
        for(j=0;j<n-i-1;j++){
            if(a[j]>a[j+1]){
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
    for(i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    return 0;
}
```

```
}
```

### 3. 语文成绩

#### 题目描述

给出 $N(5 \leq N \leq 150)$ 个人的语文成绩，求 $N$ 个人的语文总分和平均分，并按成绩高低排序后输出。

#### 输入

第1行：一个整数 $N$ 。 第2行：空格隔开的 $N$ 个整数，表示 $N$ 个人的语文成绩。

#### 输出

三行。 第1行：一个整数，为 $N$ 个人的总分。 第2行： $N$ 个人的语文平均分，保留两位小数。 第3行： $N$ 个空格隔开的整数，为从高到低输出的 $N$ 个人的成绩。

#### 样例输入

```
5
72 98 95 81 86
```

#### 样例输出

```
432
86.40
98 95 86 81 72
```

#### 示例代码

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
    int a[11],t,n,i,j,s=0;
    double avg;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
        s+=a[i];
    }
    avg=s*1.0/n;
    cout<<s<<endl;
    cout<<fixed<<setprecision(2)<<avg<<endl;
    for(i=0;i<n-1;i++){
        for(j=0;j<n-i-1;j++){
            if(a[j]<a[j+1]){
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
    for(i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    return 0;
}
```

## 4. 求中位数

### 题目描述

中位数指的是一组数，如果按照大小排序排好后最中间的那个数的值，如果有偶数个元素，那么就是最中间两个数的平均数！

比如：2 5 8 1 6，排序后的结果为1 2 5 6 8，那么这组数的中位数就是5！

再比如：8 9 1 2 3 0，排序后的结果为0 1 2 3 8 9，那么这组数的中位数就是(2+3)/2=2.5

### 输入

第一行：一个整数n代表有n个数

第二行：n个数的值

### 输出

中位数（结果保留1位小数）

### 样例输入

```
5
2 5 8 1 6
```

### 样例输出

```
5.0
```

### 思路

先对数组进行排序，并判断数组元素个数的奇偶性，随后找出中位数

n = 5 时，

下脚标：0 1 2 3 4； 中位数下脚标：2 (n/2)

n = 6 时，

下脚标：0 1 2 3 4 5； 中位数下脚标：2,3 (n/2 和 n/2-1)

### 示例代码

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
    int n,a[1000],i,j,t;
    double v;//存放中位数
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    for(i=0;i<n-1;i++){
        for(j=0;j<n-i-1;j++){
            if(a[j]>a[j+1]){
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
}
```

```
if(n%2!=0){  
    v=a[n/2];  
}  
else{  
    v=(a[n/2-1]+a[n/2])/2.0;  
}  
cout<<fixed<<setprecision(1)<<v<<endl;
```

## 2.课后作业

### 1. 寻找第K大数

#### 题目描述

N个小朋友在一起做游戏。每个小朋友在自己的硬纸板上写一个数，然后同时举起来。接着，小y老师提一个问题，看哪个小朋友先抢答出来。问题是：在这N个数中，第K大的是哪个数？请你编程完成。

#### 输入

输入文件的第一行为2个整数，依次为N和K( $K \leq N \leq 10000$ )。下面N行，每行一个整数，表示从第1个小朋友到第N个朋友分别写的数。假设这些小朋友只知道-32768~32767之间的数。

#### 输出

输出文件只有一行，就一个数，为第K大的那个数。

#### 样例输入

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

#### 样例输出

```
2
```

### 2. 优秀成绩的平均分

#### 题目描述

期中考试结束了，老师想知道这次语文考试前5名同学的平均分是多少，请你编程来帮老师计算一下！

#### 输入

第一行，一个整数n，代表本次考试的总人数 ( $5 \leq n \leq 100$ )

第二行n个整数，代表n个人的语文成绩（这n个人的分数是无序的）

#### 输出

语文成绩前5名同学的平均分（结果保留1位小数）

#### 样例输入

```
10  
98 98 100 96 99 90 91 87 80 100
```

#### 样例输出

99.0

### 3. 第K大与第K小数

#### 题目描述

给定一个长度为N( $0 < n \leq 10000$ )的序列，保证每一个序列中的数字 $a[i]$ 是正整数，编程要求求出整个序列中第k大的数字减去第k小的数字的值m，并判断m是否为质数。 $(0 < k \leq n)$

#### 输入

输入格式：第一行为2个数n, k (含义如上题) 第二行为n个数，表示这个序列

#### 输出

输出格式：如果m为质数则第一行为'YES'(没有引号) 第二行为这个数m 否则 第一行为'NO' 第二行为这个数m

#### 样例输入

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

#### 样例输出

```
YES
2
```

### 4. 学员的名次

#### 题目描述

期末考试语文成绩出来了，老师在课堂上公布了每位同学的语文成绩，小明想查一下自己的成绩在班级能排到第几名。请你编写一个程序，根据给定的所有同学的语文成绩以及小明的语文成绩，计算出小明的排名。（假设所有人成绩都不相等）

#### 输入

第一行一个整数n代表学生总人数 ( $n \leq 100$ )

第二行，有n个整数，代表n个语文成绩（这些成绩都是0~100之间的分数）

第三行一个整数x代表小明同学的语文成绩

#### 输出

一个整数，代表小明同学的名次

#### 样例输入

```
5
98 100 99 80 87
98
```

#### 样例输出

```
3
```

### 5. 需要安排几位师傅加工零件？

#### 题目描述

某工厂有n个零件加工的师傅，每位师傅每天能够加工出不同数量的零件。现有m个零件要求一天加工完，请问该工厂最少需要派几个师傅来完成这次零件加工任务，如果安排所有的师傅都参与加工也不能在一天内完成任务，请输出“NO”。 (4.2.71)

#### 输入

第一行有两个整数，用空格隔开；第一个整数代表要加工的总零件个数m ( $m \leq 10^6$ )，第二个整数代表工厂的零件加工师傅的数量n ( $n \leq 100$ )。

第二行有n个整数，分别代表每个师傅每天能够加工出来的零件数量（每个师傅每天加工的零件数量 $\leq 10^4$ ）。

#### 输出

工厂在1天时间内加工所有零件需要的师傅数量或者输出NO。

#### 样例输入

```
10 5
1 3 2 4 2
```

#### 样例输出

```
4
```

## 3.利用数组存放运算结果

### 1. COUNT

#### 题目描述

一本书的页数为N，页码从1开始编起，请你求出全部页码中，用了多少个0，1，2……9。

#### 输入

一个正整数N ( $N \leq 10000$ )，表示总的页码。

共十行：第k行为数字k-1的个数。

#### 样例输入

```
11
```

#### 样例输出

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

#### 思路

准备长度为10的数组，分别存放0~9出现的次数， $a[0]$ 存放0出现的次数， $a[1]$ 存放1出现的次数，以此类推；

循环拆出每个页码中出现的数字，每拆完一个就+1再拆，直到拆到指定数字。

### 示例代码

```
#include<iostream>
using namespace std;
int main(){
    int n,a[10]={0},i,t;
    cin>>n;
    for(i=1;i<=n;i++){
        t=i;
        while(t!=0){
            a[t%10]++;
            t/=10;
        }
    }
    for(i=0;i<10;i++){
        cout<<a[i]<<endl;
    }
}
```

## 2. 数字出现次数

### 题目描述

有50个数（0-19），求这50个数中相同数字出现的最多次数为几次？

### 输入

50个数字

### 输出

1个数字（即相同数字出现的最多次数）

### 样例输入

```
1 10 2 0 15 8 12 7 0 3 15 0 15 18 16 7 17 16 9 1 19 16 12 17 12 4 3 11 1 14 2
11 14 6 11 4 6 4 11 13 18 7 0 3 2 3 18 19 2 16
```

### 样例输出

```
4
```

### 思路

准备一个长度为20的数组，每个下脚标i对应的元素就存储数字中i出现的次数。由于数组存储的是每个数字出现的次数，因此数组的最大数就是出现次数最多的数字。

### 实现方式一

```
#include<iostream>
using namespace std;
int main(){
    //存放读入的数据
    int a[50];
    //记录数字出现的次数
    int c[20]={0};
    int i,max;//i用作循环，max记录出现次数最多数字的次数
    for(i=0;i<50;i++){
```

```

    cin>>a[i];
    c[a[i]]++;
}
max=c[0];
for(i=0;i<20;i++){
    if(max< c[i]){
        max=c[i];
    }
}
cout<<max<<endl;
return 0;
}

```

## 实现方式二

```

#include<iostream>
using namespace std;
int main(){
    //存放读入的数据
    int x;
    //记录数字出现的次数
    int c[20]={0};
    int i,max;//i用作循环， max记录出现次数最多数字的次数
    for(i=0;i<50;i++){
        cin>>x;
        c[a[i]]++;
    }
    max=c[0];
    for(i=0;i<20;i++){
        if(max< c[i]){
            max=c[i];
        }
    }
    cout<<max<<endl;
    return 0;
}

```

### 3. 去除重复数字

#### 题目描述

给你N个数 ( $n \leq 100$ ) ,每个数都在 (0~1000) 之间, 其中由很多重复的数字, 请将重复的数字只保留一个, 并将剩下的数由小到大排序并输出。

#### 输入

输入有2行,

第1行为1个正整数, 表示数的个数:N

第2行有N个用空格隔开的整数。

#### 输出

第1行为1个正整数M, 表示不相同数的个数。

接下来的M行, 每行一个整数, 表示从小到大排好序的不相同的数。

#### 样例输入

```
10
20 40 32 67 40 20 89 300 400 15
```

### 样例输出

```
15
20
32
40
67
89
300
400
```

### 思路1：

每读入一个数字，就在数组中循环一遍，看看数组中是否存在这个数，如果有就不存储，如果没有就存进去。在本题中，需要使用一个标记位来标记是否有重复的数字存在。

**bool (布尔型)** 通常用作标记真假 (true/false)

```
bool flag = true;//真
bool flag = false;//假
bool flag = 3<2;//假
bool flag = 3>2;//真
```

### 实现方式一

```
#include<iostream>
using namespace std;
int main(){
    int i,n,a[100],x,j,t;
    bool f=false;//标记是否有重复的数字
    int k=0;//记录实际上去掉重复数字后数组元素个数
    cin>>n;
    for(i=0;i<n;i++){
        cin>>x;
        f=false;//每一个数字进来都要将标记位初始化
        for(j=0;j<k;j++){
            if(x==a[j]){
                f=true;
                break;
            }
        }
        if(!f){
            a[k]=x;
            k++;
        }
    }
    for(i=0;i<k-1;i++){
        for(j=0;j<k-1-i;j++){
            if(a[j]>a[j+1]){
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
}
```

```

        }
    }
    for(i=0;i<k;i++){
        cout<<a[i]<<endl;
    }
    return 0;
}

```

### 思路2：

因为数字是从0~1000，所以可以用长度为1001的数组f来标记某个数字是否出现过，如果f[x]=0.认为没有出现过，就将x存入数组a并将f[x]标记为1。

```

#include<iostream>
using namespace std;
int main(){
    int i,n,a[100],x,j,t;
    int f[1001]={0}; //用来标记的数组
    int k=0;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>x;
        if(f[x]==0){
            f[x]=1;
            a[k]=x;
            k++;
        }
    }
    for(i=0;i<k-1;i++){
        for(j=0;j<k-1-i;j++){
            if(a[j]>a[j+1]){
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
    for(i=0;i<k;i++){
        cout<<a[i]<<endl;
    }
    return 0;
}

```

### 思路3：

事实上，采用方法二实现以后，排序可以省略，只需要稍微改动一下代码即可。同学们可以思考一下并尝试。

## 4.课后作业

### 1. 求N个整数的平均数、众数和中位数

#### 题目描述

求N个整数的平均数，众数和中位数。

小知识：

\* 众数

如有9个数：17 13 17 9 17 17 3 16 17 17出现的次数最多，即为这组数的众数。

\* 中位数

如有9个数：102 170 96 90 97 106 110 182 100

将这9个数按一定的顺序（从大到小或从小到大）排列后得到：

182 170 110 106 102 100 97 96 90 正中间的一个数是102，102是这组数的中位数。

而这10个数：106 99 104 120 107 112 33 102 97 100

按一定顺序排列后得到：120 112 107 106 104 102 100 99 97 33

正中间有两个数：104 102，中位数就是这两个数的平均数，即  $(104+102) / 2 = 103$ 。

**输入**

第一行为整数N ( $5 \leq N \leq 10000$ ) 第二行为空格隔开的N个数 $A_i$  ( $0 \leq A_i \leq 100$ )

**输出**

输出空格隔开的平均数 众数 中位数（平均数保留两位小数，中位数保留一位小数）。

**样例输入**

```
6
5 2 2 3 4 6
```

**样例输出**

```
3.67 2 3.5
```

## 2. 邮票组合

**题目描述**

某人有m张3分的邮票和n张5分的邮票，用这些邮票中的一张或若干张（也可以是0张）可以得到多少种不同的大于0的邮资？请找出可能组合的邮资方案总数，并按照由小到大的顺序输出所有不重复的大于0的方案！（5.1.97）

如：1张3分和1张5分可能的邮资组合如下

0张3分+1张5分=5分

1张3分+0张5分=3分

1张3分+1张5分=8分

因此，可能的方案有3种，排序后的结果是：3 5 8！

**输入**

两个整数，m和n，分别代表了3分和5分的邮票的数量！ ( $1 \leq m, n \leq 100$ )

**输出**

输出有两行，第一行输出这两种邮票能组合的不同的大于0的邮资方案，数与数之间用空格隔开！

第二行输出可能的方案总数！

**样例输入**

```
2 2
```

### 样例输出

```
3 5 6 8 10 11 13 16  
8
```

### 3. 扑克牌组合

#### 题目描述

小明从一副扑克牌中（没有大小王，J认为是数字11，Q是12，K是13，A是1）抽出2张牌求和，请问能够组合出多少个不相等的数，按照由小到大输出这些数。 (5.1.98)

#### 输入

第一行是一个整数n代表 (n<=52) 扑克牌的总数量

第二行的n个整数分别代表扑克牌的数值

#### 输出

第一行是一个整数m代表组合出不相等的数字个数。

第二行m个数用空格隔开代表这m个由小到大排序的不相等的数。

#### 样例输入

```
4  
3 1 2 4
```

#### 样例输出

```
5  
3 4 5 6 7
```

## 第十五章 sqrt()函数及其应用

### 1. 判断完全平方数

#### 1. 什么是平方根？

平方根，又叫二次方根，表示为：

$$\sqrt{n}$$
$$x^2 = n, \sqrt{(n)} = x$$

例题：写出下列算式的值

$$\sqrt{(100)} = \underline{\hspace{2cm}}$$

$$\sqrt{(16)} = \underline{\hspace{2cm}}$$

$$\sqrt{(25)} = \underline{\hspace{2cm}}$$

$$\sqrt{(81)} = \underline{\hspace{2cm}}$$

#### 2. 如何在C++中求平方根？

### 1. 例1:求根号n的值

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
    int n = 36;
    double r=sqrt(n);
    cout<<r;
    return 0;
}
```

### 2. 例2：判断变量n是否为平方数

**思路：**整数n开根号取整和不取整是同一个数，就是完全平方数

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n;
    cin>>n;
    double r=sqrt(n);
    if(r==(int)r){
        cout<<"是完全平方数";
    }else{
        cout<<"不是完全平方数";
    }
    return 0;
}
```

### 3. 完全平方数

#### 题目描述

一个整数n，加上100是得到的n+100一个完全平方数，在加100的基础上再加上168得到的n+100+168又是一个完全平方数，请问该数最小是多少？

#### 输入

无

#### 输出

符合条件的最小的数

#### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
/*
题意为:
i+100和i+268都是完全平方数
*/
    int i=1;
    while(sqrt(i+100)!=(int)sqrt(i+100)||sqrt(i+268)!=(int)sqrt(i+268)){
        i++;
    }
}
```

```
    cout<<i;
    return 0;
}
```

## 2.课堂案例

### 1. 判断素数

#### 题目描述

任意输入一个整数，判断它是否为素数。是的话输出“T”，不是的话输出“F”。

质数（prime number）又称素数，质数定义为在大于1的自然数中，除了1和它本身以外不再有其他因数。

#### 输入

输入只有一行，包括1个整数。

#### 输出

输出只有一行。

#### 样例输入

```
57
```

#### 样例输出

```
F
```

**思路1：**从2~n-1中挨个寻找n的因数，如果没有找到且n>1，则n是素数。

#### 示例代码：

```
#include<iostream>
using namespace std;
int main(){
    int n,i,c=0;
    cin>>n;
    for(i=2;i<n-1;i++){
        if(n%i==0){
            c++;
        }
    }
    if(c==0&&n>1){
        cout<<"T"<<endl;
    }else{
        cout<<"F"<<endl;
    }
    return 0;
}
```

**注意：**当输入的n太大时，容易产生超时

**思路2：**优化程序结构，减少循环次数，n的因数可以在2~n/2中寻找，如果找到直接终止循环

#### 示例代码：

```

#include<iostream>
using namespace std;
int main(){
    int n,i,c=0;
    cin>>n;
    for(i=2;i<=n/2;i++){
        if(n%i==0){
            c++;
            break;
        }
    }
    if(c==0&&n>1){
        cout<<"T"<<endl;
    }else{
        cout<<"F"<<endl;
    }
    return 0;
}

```

**思路3：**继续优化代码，减少循环次数，利用因数成对求解的原理，如n=100时因数为：2 50， 4 25， 5 50， 10 10，

总结可得：因数 $1 \leq$ 因数 $2$ ，如果出现 因数 $1 >$ 因数 $2$ ，则为重复求解。

推导可得：如果成对求解因数，因数1最大等于因数2，则因数1 的平方=n。

结论：一个整数n的因数对一定可以在 $2 \sim \sqrt{n}$ 的范围内求解完毕

**示例代码：**

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int n,i,c=0;
    cin>>n;
    for(i=2;i<=sqrt(n);i++){
        if(n%i==0){
            c++;
            break;
        }
    }
    if(c==0&&n>1){
        cout<<"T"<<endl;
    }else{
        cout<<"F"<<endl;
    }
    return 0;
}

```

## 2. 因子求和

### 题目描述

已知一个正整数N ( $20 \leq N \leq 800000000$ )，请你编写程序求出该数的全部因子（不包括1和n）的和。

### 输入

一个正整数n。

## 输出

一个整数代表n的因子和。

## 样例输入

24

## 样例输出

35

## 思路：

采用上题的思路，在 $2\sim\sqrt{n}$ 之间查找

## 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n,i,s=0;
    cin>>n;
    for(i=2;i<=sqrt(n);i++){
        if(n%i==0){
            if(i!=n/i){
                s=s+i+n/i;
            }else{
                s+=i;
            }
        }
    }
    cout<<s<<endl;
    return 0;
}
```

## 3.课后作业

### 1. 判奇偶求和

#### 题目描述

输入一个正整数n,如果n是奇数,则求1-n之间所有的偶数之和；如果n是偶数,则求n所有的约数之和(包括1和本身)。

#### 输入

一行，一个整数n (0<n<3010)

#### 输出

一行，一个整数。

## 样例输入

89

## 样例输出

1980

## 2. 判断质数

### 题目描述

质数是指除了1和本身之外没有其他约数的数，如7和11都是质数，而6不是质数，因为6除了约数1和6之外还有约数2和3。输入一个正整数，判断它是否为质数，如是质数则输出“Yes”，否则输出这个数的大于1的最小的约数。

### 输入

仅有一行包含一个正整数n，其中  $1 < n < 1000000$

### 输出

仅有一行，如果从输入文件读入的数是质数则输出“Yes”，否则输出这个数的大于1的最小的约数。

### 样例输入

2009

### 样例输出

7

## 3. 寻找肇事司机

### 题目描述

一辆卡车违反交通规则，撞人后逃跑。现场有三人目击事件，但都没有记住车号，只记下车号的一些特征。

甲说：牌照的前两位数字是相同的；乙说：牌照的后两位数字是相同的，但与前两位不同；丙是数学家，他说：四位的车号刚好是一个整数的平方。请根据以上线索求出车号。

### 输入

无

### 输出

输出肇事司机的车牌号码

## 4. aabb

### 题目描述

查找形如“aabb”的四位完全平方数，即前两位数字相同，后两位数字也相同。

### 输入

无

### 输出

若干行，每行一个符合条件的四位数（从小到大）。

# 第十六章 自定义函数

## 1. 函数的定义

### 1. 什么是函数？

1. 系统函数：系统已经定义好的，方便调用完成某一个具体任务的一组语句，eg：

```
double x = sqrt(n);
cout<<fixed<<setprecision(2)<<x<<endl;
```

### 2. 什么是函数：

为了实现某个具体的任务（如求平方根，获得随机数等）而封装打包在一起的语句。

3. 现实生活中，我们会将可以实现一个任务的功能打包成一个机器，如洗衣机，电饭锅，榨汁机等，我们可以将这些电器抽象成一个个可以实现具体功能的“函数”。

### 2. 为什么要定义函数？

案例：

#### 纯粹素数

纯粹素数是这样定义的：一个素数，去掉最高位，剩下的数仍为素数，再去掉剩下的数的最高位，余下的数还是素数。这样下去一直到最后剩下的个位数也还是素数。求出所有小于3000的四位的纯粹素数。

#### 输入

无

#### 输出

按从小到大的顺序输出若干个纯粹素数，每行一个。

#### 归纳：

从本题可以看出，判断是否为素数这个功能最多需要对同一个待判断的数字使用四次，但区别仅仅是数字不同，判断功能是完全一样的，是可以复用的，如果需要修改，封装后的函数显然更方便。

### 3. 函数的基本结构及定义、使用

```
返回值类型 函数名(函数参数类型1 参数名1, 函数参数类型2 参数名2, 函数参数类型3 参数名3...){  
    return 返回值;  
}
```

#### 解释：

1. 返回值即运算后得到的结果，可以根据需要决定是否需要返回值，不需要返回值，需要将返回值类型定义为void，需要注意的是一个函数只能有一个返回值，有返回值时需要写return，返回值类型为void时，不需要写return；
2. 函数参数可以为0个也可以为多个，中间用“,”间隔。在定义时的参数叫做**形式参数**（形参），在定义时并没有真实的数据，而在调用时才需要真正的数值，这个时候叫做**实际参数**（实参）。

### 4. 函数的示例

1. 定义一个没有输入参数和返回值的函数，一经调用就输出“我大意了啊，没有闪”；

2. 定义一个有输入参数没有返回值的函数，该函数要求输入一个整数，如果整数为1，则输出“闪电五连鞭”；如果整数为2，则输出“我一下就把你鼻子打姑折了”；如果整数是3，则输出“婷婷，发生啥事了？”
3. 定义一个函数，求2个整数中的最大的一个，需要参数和返回值；
4. 定义一个函数，求3个整数中的最大值，需要调用题c中定义的函数；

```
#include<iostream>
using namespace std;
//两数最大
int doublemax(int i,int j){
    int n;
    if(i>j){
        n=i;
    }else{
        n=j;
    }
    return n;
}
int triplemax(int i,int j,int k){
    int n;
    n=doublemax(i,j);
    n=doublemax(n,k);
    return n;
}
int main(){
/*
1. 调用函数时，需要什么参数就要给什么参数；
2. 函数有返回值可以直接使用也可以用同类型变量进行接收
*/
    int n=doublemax(8,3);
    cout<<n<<endl;
    return 0;
}
```

## 5. 函数调用的注意事项：

1. 不同的函数内部定义的变量函数之间互不相通（生命周期不同，也可以理解为互不相关）；
2. 定义函数的时候不需要输入参数，在调用的时候必须要给该函数输入参数（如果函数定义了需要参数的话）；
3. 返回值问题：如果一个函数完成之后会产生一个结果或我们需要产生一个结果，则需要返回这个结果，但需要注意的是每个函数只有一个返回值，函数执行到return时则**终止整个函数**；
4. 形式参数问题：如果一个函数的运算结果会因为参数的不同而产生不同的结果，则需要定义形式参数；
5. 函数的命名规则：尽量见名知意，避免胡乱命名；
6. 定义函数的通俗理解为：函数需要什么定义（留空）什么，需要什么结果return什么；
7. 调用函数通俗的理解为：需要什么给什么，返回什么拿什么

## 6. 变量的生命周期

变量的生命周期（作用范围）：变量定义时所在的最近的一对大括号以内。

**例子：**

```

int a=10;
if(a>0){
    int b = a + 1;
}else{
    int b = a - 1;
}
cout<<b;//此处的cout会报错，因为2个b仅在if和else中分别有效

```

例子：

```

#include<bits/stdc++.h>
using namespace std;
int max(int a,int b){
    int r;
    if(a>b){
        r=a;
    }else{
        r=b;
    }
    return r;
}
int main(){
    int a=1,b=2;
    max(a,b);
    cout<<r<<endl;//这里会报错，因为在这里r并没有被定义
}

```

## 2.课堂练习

### 1. 纯粹素数

#### 题目描述

纯粹素数是这样定义的：一个素数，去掉最高位，剩下的数仍为素数，再去掉剩下的数的最高位，余下的数还是素数。这样下去一直到最后剩下的个位数也还是素数。求出所有小于3000的四位的纯粹素数。

#### 输入

无

#### 输出

按从小到大的顺序输出若干个纯粹素数，每行一个。

```

#include<bits/stdc++.h>
using namespace std;
bool sushu(int n){
    bool flag = true;
    for(int i = 2;i<=sqrt(n);i++){
        if(n%i==0){
            flag = false;
            break;
        }
    }
}

```

```
    return flag;
}
int main(){
    for(int i = 1000;i<3001;i++){
        if(sushu(i)&&sushu(i%10)&&sushu(i%100)&&sushu(i%1000)){
            cout<<i<<endl;
        }
    }
    return 0;
}
```

## 3.课后作业

### 1. 奎生素数

#### 题目描述

我们定义，如果a和a+2都是素数（如5和7），那么我们就称a和a+2是一对孪生素数。请写一个程序找出2-N之间的所有孪生素数。

#### 输入

一个整数N（ $2 < N < 1000$ ）。

#### 输出

若干行，每行两个整数，即一对孪生素数。

#### 样例输入

```
10
```

#### 样例输出

```
3 5
5 7
```

### 2. 素数回文数

#### 题目描述

如果一个数从左边读和右边读都是同一个数，就称为回文数，例如686就是一个回文数。编程求10到1000内所有的既是回文数同时又是素数的自然数。

#### 输入

无

#### 输出

若干个数 每行一个

### 3. 纯粹合数

#### 题目描述

一个合数，去掉最低位，剩下的数仍是合数，再去掉剩下的数的最低位，余留下来的数还是合数，这样反复，一直到最后剩下的一位数仍是合数；我们把这样的数称为纯粹合数。求所有的三位纯粹合数。

#### 输入

无

### 输出

若干个3位数 每行一个 (从小到大)

## 4. 求完全数的个数

### 题目描述

一个正整数若等于全部因子的和，则称此数为完全数。例如：6有因子1,2,3。同时 $6=1+2+3$ ，所以6是完全数。

### 输入

一个正整数 N( $10 \leq N \leq 100000$ )

### 输出

小于等于N的完全数的个数。

### 样例输入

```
10
```

### 样例输出

```
1
```

## 5. 桐桐数

### 题目描述

桐桐很喜欢研究数字，特别喜欢研究质数。一天，桐桐发现有一些数字可以表示成两个质数相乘的形式，比如， $10=2*5$ ，2、5都是质数，所以10是一个“桐桐数”。所以桐桐决定考考你，她告诉你一个数n，请你判断n是不是“桐桐数”。

### 输入

一个数n( $1 \leq n \leq 2^{31}-1$ )。

### 输出

输出一行，如果n是一个“桐桐数”，则输出“It's a Tongtong number.”，否则输出“It's not a Tongtong number.”

### 样例输入

```
10
```

### 样例输出

```
It's a Tongtong number.
```

## 6. 念数字

### 题目描述

编一个“念数字”的程序。当你输入一个0-99之间的数后，计算机就会用汉语拼音输出这个数的“念”法。

比如：35，念出来应该是：san shi wu；16念出来应该是shi liu，0念出来应该是ling！

### 输入

一行，一个整数。

### 输出

一行，这个整数的念法

### 样例输入

```
35
```

### 样例输出

```
san shi wu
```

## 7. 求 $s=a+aa+aaa+aaaa+aa\dots a$ 的值（附加题）

### 题目描述

求 $s=a+aa+aaa+aaaa+aa\dots a$ 的值，其中a从键盘读入。比如：读入2，则 $s=2+22=24$ 。再比如：读入5， $s=5+55+555+5555+55555=61725$

### 输入

一个整数a (a在1~9的范围内)

### 输出

整数n代表这个算式的结果

### 样例输入

```
2
```

### 样例输出

```
24
```

## 8. 等差素数组（附加题）

### 题目描述

如果两个素数之和的一半仍然是一个素数，则这三个素数可以组成一个等差素数组，如 $(3+7)/2=5$ ，则 $(3, 5, 7)$ 为一个等差素数组，编程求100以内的所有等差素数组。这里列出的 $3\ 5\ 7$ 是符合题目要求的第一个等差素数组（注意：本题不考虑3个数相等的情况）。

### 输入

无

### 输出

若干行，每行3个数。空格隔开！（每行的三个数从小到大排列，先按第一个数从小到大输出等差素数组，如果第一个数相同，再按第二个数从小到大输出）

# 第十七章 递归

## 1.什么是递归

### 1. 递归，即函数的自我调用（套娃）

**解释：**当函数的定义中，其内部操作又直接或间接地出现对自身的调用，则称这样的程序嵌套定义为递归定义。递归通常把一个大型复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解，递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算，大大地减少了程序的代码量。递归的能力在于用有限的语句来定义对象的无限集合。用递归思想写出的程序往往十分简洁易懂。

**数列递归：**若一个数列的项与项之间存在关联性，则可以使用递归实现；

**原理：**如果一个函数可以求 $A(n)$ ，那么该函数就可以求 $A(n-1)$ ，就形成了递归调用；

**注意：**起始项是不要求解的，是已知条件

**例子：**在数学上，所有偶数的集合可递归地定义为：

①一个偶数与2的和是一个偶数；

②0是一个偶数；

可见，仅需两句话就能定义一个由无穷多个元素组成的集合。在程序中，递归是通过函数的调用来实现的。函数直接调用其自身，称为直接递归；函数间接调用其自身，称为间接递归。

### 2. 递归求解问题的过程

1. 找出规律
2. 函数调用自己求解前面的项
3. 交代起始项，让递归能够停止

## 2.入门案例

### 1. 定义函数，递归求解等差数列1,4,7,10,13... ...第n项的值

**递归规律：** $A(n)=A(n-1)+3$

等差数列：相邻的两项差值相等

```
num(5)=num(4)+3  
=(num(3)+3)+3  
=(num(2)+3+3)+3  
=(num(1)+3+3+3)+3
```

```
#include<bits/stdc++.h>  
using namespace std;  
/*  
求解首项为1，差值为3的等差数列的第n项的值  
*/  
int num(int n){  
    int r;  
    //交代起始项，即第一项是1，防止进入死循环  
    //同时也是递归出口  
    if(n==1){  
        r=1;
```

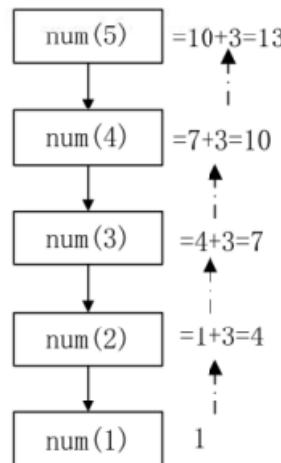
```

} else{
    r=num(n-1)+3;
}
return r;
}

int main(){
    int n;
    cin>>n;
    cout<<num(n)<<endl;
    return 0;
}

```

### 图解思路



2. 定义函数，递归求解等比数列：1,2,4,8,16... ...第n项值

**递归规律：** $A(n)=A(n-1)*2$

```

#include<bits/stdc++.h>
using namespace std;
int num(int n){
    int r;
    if(n==1){
        r=1;
    }else{
        r=num(n-1)*2;
    }
    return r;
}

int main(){
    int n;
    cin>>n;
    cout<<num(n)<<endl;
    return 0;
}

```

3. 定义函数，递归求解n的阶乘的值

**提示：** $5! = 5 \times 4 \times 3 \times 2 \times 1 = 5 \times 4!$

**规律：** $n! = (n-1)! * n \implies A(n) = A(n-1) * n$

```

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

```

```

int num(int n){
    int r;
    if(n==1){
        r=1;
    }else{
        r=num(n-1)*n;
    }
    return r;
}
int main(){
    int n;
    cin>>n;
    cout<<num(n)<<endl;
    return 0;
}

```

#### 4. 统计每个月兔子的总数

##### 题目描述

有一对兔子，从出生后第3个月起每个月都生一对兔子，一对小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问第n个月（ $n \leq 50$ ）的兔子总数为多少对？

##### 输入

输入1个整数n，表示第几个月

##### 输出

第n个月兔子的总数量有多少？

##### 样例输入

9

##### 样例输出

34

##### 思路：

第1个月： a

第2个月： a

第3个月： a b

第4个月： a b c

第5个月： a b c d e

##### 规律是什么？

$a(n)=a(n-1)+a(n-2)$  (斐波那契数列)

注意：本题有两个起始项

```

#include<bits/stdc++.h>
using namespace std;
int num(int n){
    int r;
    if(n==1 || n==2){
        r=1;
    }else{
        r=num(n-1)+num(n-2);
    }
    return r;
}

```

```

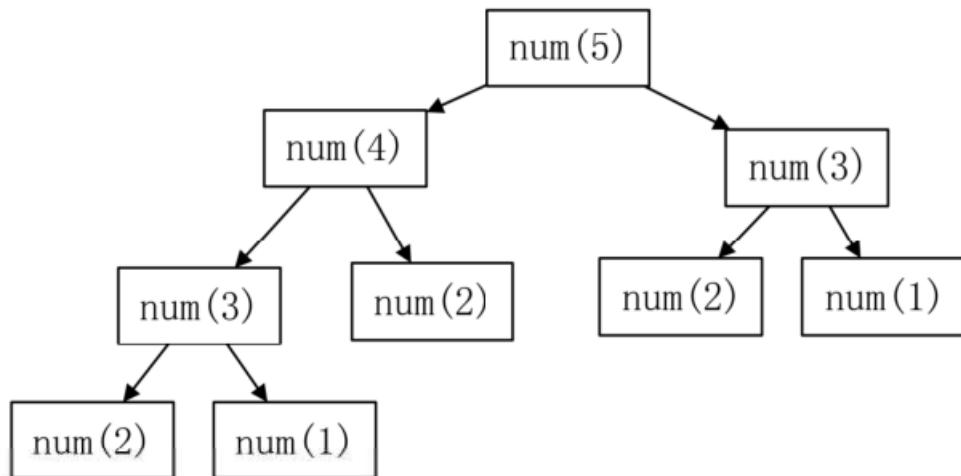
}else{
    r=num(n-1)+num(n-2);
}
return r;
}

int main(){
    int n;
    cin>>n;
    cout<<num(n)<<endl;
    return 0;
}

```

**补充:**

- 当n的值越来越大时，求解的时间也会越来越差，可能会引起超时，因为随着n的值越来越大，重复求解的项会越来越多，下图的结构会非常庞杂：



- int类型的数据最多只能表示  $2^{32}-1$ ,10位的整数，如果超出这个范围就需要使用long long类型来表达，而long long也有表达的范围极限：  $2^{63}-1$ ,约20位的整数

### 3.递归存在的问题及解决方式

#### 1. 递归存在的问题

由于递归是函数对自己的调用，因此在n的值比较大时，存在太多重复求解的过程，而这些重复求解的过程必须要等到所有项计算结束才能一层层向上汇总得出结果，因此效率非常低。

#### 2. 提高递归效率的方法

##### 1. 数组

采用数组来记录每个月兔子的数量，只需要按需求计算一次即可得到所有数据，在求解第n个月的兔子的数量的时候， $a[n-1]$ 和 $a[n-2]$ 就无需重复求解了

1	1	2	3	5		
---	---	---	---	---	--	--

**示例代码**

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int n;
    long long a[51];
    cin>>n;
    a[0]=a[1]=1;
    for(int i=2;i<n;i++){
        a[i]=a[i-1]+a[i-2];
    }
    cout<<a[n-1]<<endl;
    return 0;
}

```

## 2. 递推

我们用x和y分别标记第n-1和第n-2个月的兔子数量

1	1	2 x	3 y	5 n					
---	---	-----	-----	-----	--	--	--	--	--

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    long long x,y,n,s;
    cin>>n;
    x=1;
    y=1;
    for(int i=3;i<=n;i++){
        s=x+y;
        x=y;
        y=s;
    }
    if(n==1 || n==2){
        cout<<1<<endl;
    }else{
        cout<<s<<endl;
    }
}

```

## 4.课堂练习

### 1. 求S的值

#### 题目描述

求 $S=1+2+4+7+11+16\dots$ 的值刚好大于等于5000时S的值。

#### 输入

无

#### 输出

一行，一个整数。

#### 思路：

先找出规律：  $A(1)=1, A(2)=A(1)+1, A(3)=A(2)+2\dots \dots A(n)=A(n-1)+n-1$

```

#include<bits/stdc++.h>
using namespace std;
int num(int n){
    int r;
    if(n==1){
        r=1;
    }else{
        r=num(n-1)+n-1;
    }
    return r;
}
int main(){
    int i=1,s=0;
    while(s<5000){
        s+=num(i);
        i++;
    }
    cout<<s<<endl;
}

```

2. 求 $1/1+1/2+2/3+3/5+5/8+8/13+13/21\dots$ 的前n项的和

#### 题目描述

求 $1/1+1/2+2/3+3/5+5/8+8/13+13/21+21/34\dots$ 的前n项的和。

#### 输入

第1行：一个整数n ( $1 \leq n \leq 30$ )。

#### 输出

一行：一个小数，即前n项之和（保留3位小数）。

#### 样例输入

20

#### 样例输出

12.660

#### 思路：

找出规律，数字规律为斐波那契数列

```

#include<bits/stdc++.h>
using namespace std;
int num(int n){
    int r;
    if(n==1 || n==2){
        r=1;
    }else{
        r=num(n-1)+num(n-2);
    }
    return r;
}
int main(){

```

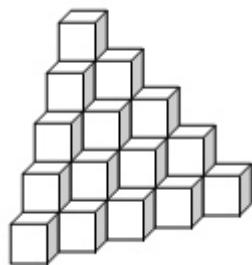
```
int n;
double s=0;
cin>>n;
for(int i=1;i<=n;i++){
    s+=num(i)*1.0/num(i+1);
}
cout<<fixed<<setprecision(3)<<s<<endl;
```

## 5.课后作业

### 1. 数数小木块

#### 题目描述

在墙角堆放着一堆完全相同的正方体小木块，如下图所示：



因为木块堆得实在是太有规律了，你只要知道它的层数就可以计算所有木块的数量了。

#### 输入

只有一个整数  $n$ ，表示这堆小木块的层数，已知  $1 \leq n \leq 100$ 。

#### 输出

只有一个整数，表示这堆小木块的总数量。

#### 样例输入

```
5
```

#### 样例输出

```
35
```

### 2. 数列求和

#### 题目描述

有一数列如下： 1 2 4 7 11 16 22..... 试求该数列前N项之和。

#### 输入

一个整数  $N$  ( $0 < N < 1000$ )。

#### 输出

一个整数。

#### 样例输入

```
6
```

## 样例输出

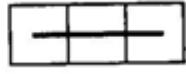
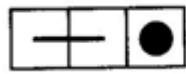
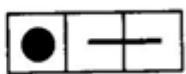
41

### 3. 骨牌铺方格

#### 题目描述

有 $1 \times n$  ( $n \leq 50$ ) 的一个长方形，用一个 $1 \times 1$ 、 $1 \times 2$ 和 $1 \times 3$ 的骨牌铺满方格，请问有多少种铺法？

例如当 $n=3$ 时为 $1 \times 3$ 的方格。此时用 $1 \times 1$ 、 $1 \times 2$ 和 $1 \times 3$ 的骨牌铺满方格，共有四种铺法。如下图：



#### 输入

一个整数 $n$  ( $n \leq 50$ )

#### 输出

骨牌的铺法

#### 样例输入

3

## 样例输出

4

# 第十八章 字符型数据

## 1. 什么是字符

char类型：字符类型，可以存储一个字符，字符在内存中以ASCII码的形式进行存储（占1个字节，8位）。即**字符的本质是ASCII编码（数字）**

ASCII 字符代码表 —

高四位		ASCII非打印控制字符												ASCII 打印字符														
		0000						0001						0010			0011			0100			0101			0110		
		0		1		2		3		4		5		6		7												
低四位	+逆制	字符	ctrl	代码	字符解释	+逆制	字符	ctrl	代码	字符解释	+逆制	字符	+逆制	字符	+逆制	字符	+逆制	字符	+逆制	字符	+逆制	字符	+逆制	字符	ctrl			
0000 0 0	BLANK NULL	^@	NUL	空	16	►	^P	DLE	数据链路转意	32	48	0	64	@	80	P	96	`	112	p								
0001 1 1	(?)	^A	SOH	头标开始	17	◀	^Q	DC1	设备控制 1	33	49	1	65	A	81	Q	97	a	113	q								
0010 2 2	(?)	^B	STX	正文开始	18	↑	^R	DC2	设备控制 2	34	50	2	66	B	82	R	98	b	114	r								
0011 3 3	(?)	^C	ETX	正文结束	19	!!	^S	DC3	设备控制 3	35	51	3	67	C	83	S	99	c	115	s								
0100 4 4	(◆)	^D	EOT	传输结束	20	¶	^T	DC4	设备控制 4	36	52	4	68	D	84	T	100	d	116	t								
0101 5 5	(♣)	^E	ENQ	查询	21	ƒ	^U	NAK	反确认	37	53	5	69	E	85	U	101	e	117	u								
0110 6 6	(♠)	^F	ACK	确认	22	■	^V	SYN	同步空闲	38	54	6	70	F	86	V	102	f	118	v								
0111 7 7	(●)	^G	BEL	震铃	23	↕	^W	ETB	传输块结束	39	55	7	71	G	87	W	103	g	119	w								
1000 8 8	(▀)	^H	BS	退格	24	↑	^X	CAN	取消	40	(	56	8	72	H	88	X	104	h	120	x							
1001 9 9	(○)	^I	TAB	水平制表符	25	↓	^Y	EM	媒体结束	41	)	57	9	73	I	89	Y	105	i	121	y							
1010 A 10	(▣)	^J	LF	换行/新行	26	→	^Z	SUB	替换	42	*	58	:	74	J	90	Z	106	j	122	z							
1011 B 11	(○)	^K	VT	竖直制表符	27	←	^[	ESC	转意	43	+	59	;	75	K	91	[	107	k	123	{							
1100 C 12	(♀)	^L	FF	换页/新页	28	█	^\`	FS	文件分隔符	44	,	60	<	76	L	92	\`	108	l	124								
1101 D 13	(♪)	^M	CR	回车	29	↔	^]	GS	组分隔符	45	-	61	=	77	M	93	]	109	m	125	}							
1110 E 14	(♫)	^N	SO	移出	30	▲	^6	RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~							
1111 F 15	(♪)	^O	SI	移入	31	▼	^-	US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	Δ	Back space						

注：表中的ASCII字符可以用：ALT + “小键盘上的数字键” 输入。

## 2.入门案例

1. 定义一个大写字母，请获取这个大写字母对应的编码，利用编码来规律求解对应的小写字母

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    //定义char类型的变量用来接收键盘得到的大写字母
    char c;
    cin>>c;
    //用int类型的n来得到c的ASCII码
    int n = c;
    //用char来接收比大写字母大32的小写字母的ASCII码
    char c2=n+32;
    //输出
    cout<<n<<" "<<c2<<endl;
}
```

注：

1. 字符的本质是ASCII编码（数字）

2. 必须记住的字符编码：'A'\==》65 'a'\==》97 '0'\==》48

2. 键盘读入一个字母，如果是大写就转为小写，如果是小写就转为对应大写。

方法一：转换得到输入字母的ASCII码并由此判断大小写

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char c, ch;
    cin>>c;
    int n = c;
    if(n>=65&&n<=90){
        ch=n+32;
    }else if(c>='A'&&c<='Z'){
        ch=n-32;
    }
    cout<<ch;
}
```

方法二：因为字符的本质是ASCII码，所以可以直接判断：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char c;
    cin>>c;
    if(c>='a'&&c<='z'){
        c-=32;
    }else if(c>='A'&&c<='Z'){
        c+=32;
    }
    cout<<c;
}
```

### 3. 转义字符

```
char a = '\'';//字符 '
char b = '\"';//字符"
char c = '\n';//换行
char d = '\\';//字符 \
cout<<a<<b<<c<<d;
```

### 4. 课堂练习

#### 1. 打印小写字母表

##### 题目描述

把英文字母表的小写字母按顺序和倒序打印出来。(每行13个)

##### 输入

无

##### 输出

输出四行

##### 样例输出

```
abcdefghijklm
nopqrstuvwxyz
zyxwvutsrqpon
mlkjihgfedcba
```

解法一：利用ASCII码的顺序进行打印

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j=0;
    char c;
    for(i=97;i<=122;i++){
        c=i;
        cout<<c;
        j++;
        if(j%13==0){
            cout<<endl;
        }
    }
    for(i=122;i>=97;i--){
        c=i;
        cout<<c;
        j++;
        if(j%13==0){
            cout<<endl;
        }
    }
}
```

解法二：直接利用字符进行比较、运算

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int j=0;
    for(char c='a';c<='z';c++){
        cout<<c;
        j++;
        if(j%13==0) cout<<endl;
    }
    for(char c='z';c>='a';c--){
        cout<<c;
        j++;
        if(j%13==0) cout<<endl;
    }
}
```

## 2. 字符图形10-字母三角

### 题目描述

输入一个整数打印字符图形

## 输入

一个整数 (0 < N < 10)

## 输出

一个字符图形

### 样例输入

```
3
```

### 样例输出

```
A  
BBB  
CCCCC
```

**思路：**类似嵌套循环中的要求，行列相关联即可。可先打印一个数字塔，然后再进行修改

规律为：第一行一个，第二行三个，第三行五个...

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n;
    char c;
    cin>>n;
    for(int i=1;i<=n;i++){
        c=i+64;
        for(int j=0;j<n-i+1;j++){
            cout<<" ";
        }
        for(int j=0;j<2*i-1;j++){
            cout<<c;
        }
        cout<<endl;
    }
}
```

**补充：**可以用(**数据类型**)**数据**进行数据类型强制转化，如上述代码中的输出位置可以修改为(**char**)**(i+64)**，因为要先计算出当前字母对应的ASCII码，所以先加运算，然后再转化为char，参考代码如下：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n;
//    char c;
    cin>>n;
    for(int i=1;i<=n;i++){
//        c=i+64;
        for(int j=0;j<n-i+1;j++){
            cout<<" ";
        }
        for(int j=0;j<2*i-1;j++){

```

```
//           cout<<c;
            cout<<(char)(i+64);
        }
        cout<<endl;
    }
}
```

## 5.课后作业

### 1. 字符图形11-字母正三角

#### 题目描述

输入一个整数打印字符图形

#### 输入

一个整数 ( $0 < N < 10$ )

#### 输出

一个字符图形

#### 样例输入

4

#### 样例输出

```
A
ABC
ABCDE
ABCDEFG
```

### 2. 打印字母塔

#### 题目描述

输入行数N,打印图形.

#### 输入

输入只有一行, 包括1个整数。( $N \leq 15$ )

#### 输出

输出有N行.

#### 样例输入

3

#### 样例输出

```
A
BAB
CBABC
```

# 第十九章 字符型数组

## 1.什么是字符型数组

### 1. 字符型数组

用来存储字符的数组，也可以称为“字符串”。

### 2. 字符数组的特点

以字符'\0'结尾

例如：

```
char a = '\0'; //空字符，其实并不存在这个字符，ASCII码是0
char b = ' '; //空格字符，ASCII码是32
int x=a,y=b;
cout<<x<<endl<<y;
```

### 3. 试一试

仿照定义整数数组的形式，定义字符数组并为字符数组赋值

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    //形式一： 定义字符数组但不赋值（必须以字符‘\0’结尾）
    char ch[10];
    ch[0]='c';
    ch[1]='o';
    ch[2]='d';
    ch[3]='e';
    cout<<ch<<" "<<sizeof(ch)<<" "<<strlen(ch)<<endl;
    //形式二： 定义数组并为数组赋值（需要注意的是，如果为数组赋值的值不足数组的长度，则余下的数组元素的值默认为'\0'）
    //sizeof() 用来计算变量占用的字节数量（一个字符占1个字节）
    //strlen() 用来计算字符串实际占用的字符数量（\0之前有多少字符）
    char b[10]={'h','i','\0'};
    cout<<b<<" "<<sizeof(b)<<" "<<strlen(b)<<endl;
    //形式三： 定义数组，长度由内容决定
    char c[]={'h','i','\0'};
    cout<<c<<" "<<sizeof(c)<<" "<<strlen(c)<<endl;
    //形式四： 直接定义字符数组为字符串
    char d="coderbee";
    cout<<d<<" "<<sizeof(d)<<" "<<strlen(d)<<endl;
    char e[10]={'h','e','l','l','o','\0','h','i'};
    cout<<e<<" "<<sizeof(e)<<" "<<strlen(e)<<endl;
}
```

a 数组	h	i	\0	随机	随机	随机	随机	随机	随机	
下标	0	1	2	3	4	5	6	7	8	9
b 数组	h	i	\0	\0	\0	\0	\0	\0	\0	
下标	0	1	2	3	4	5	6	7	8	9
c 数组	h	i	\0							
下标	0	1	2							
d 数组	h	i	\0							
下标	0	1	2							
e 数组	h	e	l	l	o	\0	h	i	\0	\0
下标	0	1	2	3	4	5	6	7	8	9

#### 4. 字符数组与整数数组的区别

1. 字符数组存储字符而整数数组存储整数
2. 字符数组必须以'\0'结尾，作为字符数组结束的标记
3. 字符数组因为以'\0'结尾，所以可以直接输出，而直接输出字符数组只能得到地址
4. 字符数组可以直接输入（cin或gets），整数数组需要逐个输入

#### 5. 键盘为字符数组赋值并遍历

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char s[100];
    // cin之后，字符数组默认用'\0'补齐
    //*****特点：以' '结束，也就是说如果字符中出现空格默认输入内容结束，
    //但是这里仅限于控制台输入的时候有效，因为空字符和空格在ASCII码中对应的内容是不一样的！！！
    //\0是空字符，空格是32
    // cin>>s;
    // gets()以回车作为结束
    gets(s);
    cout<<s<<endl;
    // 用整数数组的遍历方式遍历字符数组
    for(int i=0;i<strlen(s);i++){
        cout<<s[i];
    }
    // 或者依据字符数组以'\0'结束的特点，将for循环写作for(int i=0;a[i]!='\0';i++)
    return 0;
}
```

## 【特别补充】

使用cin.getline()替换gets的说明：

由于信息学竞赛升级到C++14.所以gets函数不再可用，可以用cin.getline()来替换gets，下面所有出现gets的地方请大家使用cin.getline()进行替换。

**格式：** cin.getline(字符数组，字符个数n, 结束字符C)

**含义：**读入最多N-1个字符，直到遇到指定的字符C，如果没有指定结束字符，则遇到回车符结束。

**使用注意：**数组要定义比最大大小+10，防止不够存，给第2个字符个数时，要比题目要求的字符数量的上限至少多给1.

**例子：**

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    //如果题目描述告知字符数量最多100个，至少要定义101个
    //多一个字符存结束字符，但为了避免出错，建议多开10个
    char s[110];
    cin.getline(s, 5);
    //分别读入5个字符和6个字符查看结果并比较
    cout<<s<<endl;
    //定义后默认给自己预留一位用来存储换行符，所以最多存n-1个
    //所以一般至少要多写1个，以本题为例，为了防止出错可以写成 cin.getline(s,105);
    cin.getline(s, 20, 'a');//最多读入19个，以'a'结尾不包含'a'，遇到换行也不结束

    return 0;
}
```

## 2.课堂练习

### 1. 统计字符的个数

#### 题目描述

从键盘中任意输入一串字符,直至输入"#"字符代表结束.请编程统计输入的字符中的大写字母,小写字母和数字字符的个数分别是多少?

#### 输入

输入只有一行,包括一串字符.(长度小于20)

#### 输出

输出只有一行 (这意味着末尾有一个回车符号) , 包括3个整数。分别代表大写字符, 小写字符和数字字符的个数。

#### 样例输入

```
daDSALDdcada3240#
```

#### 样例输出

5 7 4

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char str[21];
    int d=0,x=0,s=0;
    cin>>str;
    for(int i=0;i<strlen(str);i++){
        if(str[i]>='A'&&str[i]<='Z'){
            d++;
        }else if(str[i]>='a'&&str[i]<='z'){
            x++;
        }else if(str[i]>='0'&&str[i]<='9'){
            s++;
        }
    }
    cout<<d<<" "<<x<<" "<<s;
}
```

## 2. 数字和

### 题目描述

输入一个很大的数，求各位上的数字和。

### 输入

一个很大的整数（不超过200位）

### 输出

一个整数

### 样例输入

```
123
```

### 样例输出

```
6
```

```
/*
```

思路：按照字符串的形式读入，然后挨个取出并转化为数字，0的ASCII码为48，以此类推  
\*/

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char str[201];
    int s=0;
    cin>>str;
    for(int i=0;i<strlen(str);i++){
        s+=(str[i]-48);
    }
    cout<<s;
}
```

**注:** str[i]为char类型数据

### 3. 调换位置

#### 题目描述

将用逗号隔开的两个英语单词交换位置输出。

#### 输入

一行以逗号隔开的两个英文单词

#### 输出

将两个单词交换后输出的结果

#### 样例输入

```
abc , de
```

#### 样例输出

```
de , abc
```

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char str[100],str1[100];
    int f,i,j=0;
    gets(str);
    //先找到逗号的位置
    for(i=0;i<strlen(str);i++){
        if(str[i]==','){
            f=i;
            break;
        }
    }
    //将逗号之后的数据存起来
    for(i=f+1;i<strlen(str);i++){
        str1[j]=str[i];
        j++;
    }
    //将逗号存起来
    str1[j]=',';
    j++;
    //将剩下的存起来
    for(i=0;i<f;i++){
        str1[j]=str[i];
        j++;
    }

}
cout<<str1;
}
```

### 4. 判断是否构成回文

#### 题目描述

输入一串字符,字符个数不超过100,且以"."结束。 判断它们是否构成回文。

## 输入

输入只有一行，包括一串字符。

## 输出

输出只有一行.TRUE 或者FALSE

### 样例输入

```
12321.
```

### 样例输出

```
TRUE
```

**思路：**判断对称位置是否相等，只要有一个不同就不是。

1. 循环下脚标[0,(strlen(s)-1)/2]找到'.'的位置f;

2. 归纳规律：

0-->f-1;

1-->f-2;

2-->f-3;

规律为：下脚标为i的元素对应的是f-1-i;

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char s[101];
    int f,i;
    bool r=true;
    gets(s);
    for(i=0;i<strlen(s);i++){
        if(s[i]=='.'){
            f=i;
        }
    }
    for(i=0;i<(f-1)/2;i++){
        if(s[i]!=s[f-i-1]){
            r=false;
            break;
        }
    }
    if(r){
        cout<<"TRUE";
    }else{
        cout<<"FALSE";
    }
    return 0;
}
```

## 5. 统计字母出现次数

## 题目描述

输入一串小写字母（以“.”为结束标志），统计出每个字母在该字符串中出现的次数(若某字母不出现，则不要输出，题目保证每个字母出现的次数<10)。

## 输入

输入只有一行，包括若干个字符。

## 输出

输出只有两行，第一行为出现的小写字母，第二行为字母的出现次数。

## 样例输入

```
abdceeff.
```

## 样例输出

```
abcdef  
111131
```

## 思路

1. 先用gets()读取键盘输入并找到'.'的位置；
2. 准备一个长度为26的int类型的数组并初始化为0，对应的下脚标+97即可存储对应的字符；
3. 逐个判断字符并将其减掉97存到对应的计数数组下脚标中；
4. 输出字符的时候判断计数单元是否为0，是0就不输出，不是0就+97转化成字符再输出；
5. 输出次数的逻辑与第四条类似

## 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char s[260];
    int c[26]={0};
    int f,i,j;
    gets(s);
    for(i=0;i<strlen(s);i++){
        if(s[i]=='.'){
            f=i;
        }
    }
    for(i=0;i<f-1;i++){
        c[s[i]-97]++;
    }
    for(i=0;i<26;i++){
        if(c[i]!=0){
            cout<<char(i+97);
        }
    }
    cout<<endl;
    for(i=0;i<26;i++){
        if(c[i]!=0){
            cout<<c[i];
        }
    }
}
```

```
    }
}
return 0;
}
```

## 6. 国王的魔镜

### 题目描述

国王有一个魔镜，可以把任何接触镜面的东西变成原来的两倍——只是，因为是镜子嘛，增加的那部分是反的。比如一条项链，我们用AB来表示，不同的字母表示不同颜色的珍珠。如果把B端接触镜面的话，魔镜会把这条项链变为ABBA。如果再用一端接触的话，则会变成ABBAABBA（假定国王只用项链的某一端接触魔镜）。给定最终的项链，请编写程序输出国王没使用魔镜之前，最初的项链可能的最小长度。

输入

只有一个字符串，由大写英文字母组成，表示最终的项链。

输出

只有一个整数，表示国王没使用魔镜前，最初的项链可能的最小长度。

样例输入

```
ABBAABBA
```

样例输出

```
2
```

示例代码

```
#include<bits/stdc++.h>
using namespace std;
//定义函数判断是否为偶数回文，因为镜面只能翻倍且前后对应
bool judge(char c[]){
    if(strlen(c)%2==0){
        for(int i=0;i<strlen(c)/2;i++){
            if(c[i]!=c[strlen(c)-i-1]){
                return false;
            }
        }
        return true;
    }
    return false;
}
int main(){
    char c[200];
    cin>>c;
    while(judge(c)){
        //只要在长度一半的位置存入一个'\0'即可实现数组去掉一半
        c[strlen(c)/2]='\0';
    }
    cout<<strlen(c);
    return 0;
}
```

### 3.课后作业

#### 1. 大小写转换

##### 题目描述

把一个字符串里所有的大写字母换成小写字母，小写字母换成大写字母。其他字符保持不变。

##### 输入

输入为一行字符串，其中不含空格。长度不超过80个字符。

##### 输出

输出转换好的字符串。

##### 样例输入

```
ABCDefgh123
```

##### 样例输出

```
abcdEFGH123
```

#### 2. 扫描识别

##### 题目描述

“扫描识别”你知道是怎么回事吧？它的意思就是：先用扫描仪把纸上的文字扫描成一个图片，再用识别软件把那个图片中的文字识别出来，最后生成一个文本文件。这对于需要把大量的纸稿录入成电子文档的人来说，当然是非常方便的。以现有的技术，扫描效果是比较理想的，但识别效果还不十分令人满意，经常会出现错误，尤其是当两个字形状特别接近的时候，而且，这种错误是很难用眼睛看出来的。我们的纸稿上有一个数字串，识别之后得到的字符串保存在输入文件中，这个串可能有识别错误。已知，可能出现的错误有如下几种： 1、把数字0错误地识别为大写字母O； 2、把数字1错误地识别为小写字母l； 3、把数字2错误地识别为大写字母Z； 4、把数字5错误地识别为大写字母S； 5、把数字6错误地识别为小写字母b； 6、把数字8错误地识别为大写字母B； 7、把数字9错误地识别为小写字母q。你的改正方案是：如果字符串中出现了上述字母，请替换为原来的数字。最后把改正之后的数字串输出。

##### 输入

只有一个字符串，表示识别后得到的字符串。串的长度不超过100。

##### 输出

只有一个数字串，表示改正后的数字串。

##### 样例输入

```
3211088BqS
```

##### 样例输出

```
3211088895
```

#### 3. 删除指定字符

##### 题目描述

从键盘输入一个字符串str和一个字符c，删除str中的所有字符c并输出删除后的字符串str。

#### 输入

第一行是一个字符串；（不含空格）

第二行是一个字符。

#### 输出

删除指定字符后的字符串。

#### 样例输入

```
sdf$$$sdf$  
$
```

#### 样例输出

```
sdfsdf
```

### 4. 倒置输出字符串

#### 题目描述

随机输入一个长度不超过255的字符串，将其倒置后输出。

#### 输入

只有一行。

#### 输出

只有一行。

#### 样例输入

```
asdfghjk1123456
```

#### 样例输出

```
6543211kjhgfdsa
```

### 5. 字符统计

#### 题目描述

输入一串小写字母(以'.'为结束标志)，统计出每个字母在该字符串中出现的次数(若某字母不出现，则不要输出)。要求：每行输出5项，每项以空格隔开。

#### 输入

输入一行以'.'结束的字符串

#### 输出

输出相应小写字母的个数。

#### 样例输入

```
aaaabbbccc.
```

#### 样例输出

a:4 b:3 c:3

## 6. 时间的差!

### 题目描述

看到两个标准格式的时间，有小时，有分钟，有秒，格式如：h:m:s，即 时:分:秒 你想知道，这两个时间之间相差多少吗？

### 输入

输入包括两行，两行均为一个“时:分:秒”格式的时间。且本题保证第一个时间一定大于第二个时间！

### 输出

输出就是输入的两个时间之间的秒数差。

### 样例输入

```
01:10:10  
00:30:30
```

### 样例输出

```
2380
```

## 4. 附加题

### 1. 简单加密

#### 题目描述

Julius Caesar曾经使用过一种很简单的密码。对于明文中的每个字符，将它用它字母表中后5位对应的字符来代替，这样就得到了密文。比如字符A用F来代替。如下是密文和明文中字符的对应关系。

密文：A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

明文：V W X Y Z A B C D E F G H I J K L M N O P Q R S T U

你的任务是对给定的密文进行解密得到明文。

你需要注意的是，密文中出现的字母都是大写字母。密文中也包括非字母的字符，对这些字符不用进行解码。

### 输入

一行，给出密文，密文不为空，而且其中的字符数不超过200。

### 输出

输出一行，即密文对应的明文。

### 样例输入

```
NS BFW, JAJSYX TK NRUTWYFSHJ FWJ YMJ WJXZQY TK YWNANFQ HFZXJX
```

### 样例输出

```
IN WAR, EVENTS OF IMPORTANCE ARE THE RESULT OF TRIVIAL CAUSES
```

### 2. 字符串的反码

#### 题目描述

一个二进制数，将其每一位取反，称之为这个数的反码。下面我们定义一个字符的反码。如果这是一个小写字符，则它和字符'a'的距离与它的反码和字符'z'的距离相同；如果是一个大写字符，则它和字符'A'的距离与它的反码和字符'Z'的距离相同；如果不是上面两种情况，它的反码就是它自身。举几个例子，'a'的反码是'z'；'c'的反码是'x'；'W'的反码是'D'；'1'的反码还是'1'；'/'的反码还是'/'。

一个字符串的反码定义为其所有字符的反码。我们的任务就是计算给出定字符串的反码。

### 输入

一个长度不超过80个字符的字符串。

### 输出

字符串的反码。

### 样例输入

```
He11o
```

### 样例输出

```
Svoo1
```

## 3. 字符串加密？

### 题目描述

在情报传递过程中，为了防止情报被截获，往往需要对情报用一定的方式加密，简单的加密算法虽然不足以完全避免情报被破译，但仍然能防止情报被轻易的识别。我们给出一种最简的的加密方法，对给定的一个字符串，把其中从a-y，A-Y的字母用其后继字母替代，把z和Z用a和A替代，其他非字母字符不变，则可得到一个简单的加密字符串。

### 输入

输入一行，包含一个字符串，长度小于80个字符。

### 输出

输出每行字符串的加密字符串。

### 样例输入

```
Hello! How are you!
```

### 样例输出

```
Ifmmp! Ipx bsf zpv!
```

## 4. 看完动漫要几天？

### 题目描述

一部完整的动漫共有 $m$ 分钟 ( $m \leq 1000$ )。为了保护视力，妈妈决定让小明同学每天从 $xx:xx$ 分看到 $xx:xx$ 分，请问小明同学需要几天才能看完这部完整的动漫。 (4.1.15)



### 输入

3行，第一行是一个整数m代表动漫的总分钟数；第二行表示每天看动漫的开始时间，第三行表示每天看动漫的结束时间。（确保输入的开始时间 < 结束时间）

### 输出

一个整数，代表小明看完动漫至少需要花的总天数。

### 样例输入

```
288  
17:00  
17:32
```

### 样例输出

```
9
```

## 5. 时钟旋转 (2)

### 题目描述

时钟从时间：xx:xx（xx时xx分），走到时间：xx:xx（xx时xx分），时针共旋转了多少度？（假设第一个时间<=第二个时间，2个时间都是12小时制，且两个时间的时间差不超过12小时，也就是说时针旋转的度数在360度之内，也就是 $1 \leq \text{时间}1 \leq \text{时间}2 \leq 12$ ）



### 输入

2行，第一行为起始时间（如：01:00），第二行为结束时间（如：01:05）

### 输出

时针旋转的度数（结果保留1位小数）

### 样例输入

01:00

01:05

### 样例输出

2.5

## 6. 贝贝的车牌问题

### 题目描述

广州市车管所为每一辆入户的汽车都发放一块车牌，车牌的号码由六个字符组成，如A99452、B88888等，这个字符串从左边数起的第一个字符为大写英文字母，如A、B、C等，表示这辆车是属于广州市区内的汽车还是郊区的汽车，后面的五位由数字组成。假定以字母A、B、C、D、E、F、G、R、S、T开头的表示是市区车牌，而以其他字母开头的表示郊区车牌。

车管所把这个任务交给贝贝。请你帮贝贝找出所给出的车牌中有多少辆是广州郊区的汽车。

### 输入

第1行是一个正整数N ( $1 \leq N \leq 10^6$ )，表示共有N个车牌。接下来的N行，每行是一个车牌号。题目保证给出的车牌不会重复。

### 输出

只有1行，即广州郊区车牌的数量。

### 样例输入

```
3
G54672
Q87680
P77771
```

### 样例输出

2

## 7. 求无暇素数

### 题目描述

一个两位整数A本身是素数，若将其个位数字与十位数字交换，得到一个新的两位数B，而B也是素数，我们则称A为无暇素数。例如：31是素数，个位数字与十位数字交换后得到13，也是素数。所以31是无暇素数。问题：给出一个数字字符串（即字符串中的字符全部由数字组成），求出其所有组成的无暇素数。

### 输入

一个数字字符串（长度 $\leq 20$ ），以“%”结束。

### 输出

全部的无暇素数，之间用一个逗号隔开。

### 样例输入

321314%

### 样例输出

# 第二十章 字符数组的进阶问题

## 1.字符串缩写、压缩问题

### 1. 词组缩写

#### 题目描述

定义：一个词组中每个单词的首字母的大写组合称为该词组的缩写。 比如，C语言里常用的EOF就是end of file的缩写。

#### 输入

测试数据占一行，有一个词组，每个词组由一个或多个单词组成；每组的单词个数不超过10个，每个单词有一个或多个大写或小写字母组成； 单词长度不超过10，由一个或多个空格分隔这些单词。

#### 输出

输出规定的缩写

#### 样例输入

```
end of file
```

#### 样例输出

```
EOF
```

#### 思路：

##### 1.获取每个单词的首字母

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char s[200];
    gets(s);
    for(int i=0;i<strlen(s);i++){
        //第一个字符不是空格就是首字母
        if(i==0&&s[i]!=' '){
            cout<<s[i];
        }
        //如果不是第一个字符，且前面是空格但自己不是空格，那么就是首字母
        }else if(s[i-1]==' '&&i!=0&&s[i]!=' '){
            cout<<s[i];
        }
    }
    return 0;
}
```

##### 2.判断大小写

```

#include<bits/stdc++.h>
using namespace std;
//定义一个函数专门负责大小写转化
char change(char c){
    if(c>='a'&&c<='z'){
        c-=32;
    }
    return c;
}

int main(){
    char s[200];
    gets(s);
    for(int i=0;i<strlen(s);i++){
        //如果是第一个单词
        if(i==0&&s[i]!=' '){
            cout<<change(s[i]);
        //如果是其它单词
        }else if(s[i-1]==' '&&i!=0&&s[i]!=' '){
            cout<<change(s[i]);
        }
    }
    return 0;
}

```

## 2. 字符串压缩

### 题目描述

输入字符串，输出压缩后的字符串。压缩的方法是把连续的相同字母压缩为"长度+字母"的形式，在本题中，单个的字母不需要压缩。

### 输入

一行，一个字符串,只包含小写英文字母,长度不超过255。

### 输出

### 样例输入

```
aaabbbbbb
```

### 样例输出

```
3a5bx
```

### 思路：

切记要仔细审题，题目要求的是压缩，如aaabba压缩的结果应该是3a2ba，而不是4a2b。

找一个计数器，只要当前这个跟下一个字符一样那就持续+1，如果不等了那就进入下一个判断：如果计数器只有1那么直接输出字符，否则输出计数+字符，随后计数器清零。

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    char s[300];
    int c=0;

```

```

gets(s);
for(int i=0;i<strlen(s);i++){
    c++;
    if(s[i]!=s[i+1]){
        if(c==1){
            cout<<s[i];
        }else{
            cout<<c<<s[i];
        }
        c=0;
    }
}
return 0;
}

```

## 2.字符串数组相关函数及练习

字符串函数	函数含义	备注
strlen(s)	字符串长度	返回字符串实际长度
strcmp(s1,s2)	字符串比较，按照字典顺序比较大小	如果s1的字典码大返回整数，相等返回0，否则返回负数
strncmp(s1,s2,n)	把s1,s2的前n个数字进行比较	
strcat(s1,s2)	将s2连接到s1后面	注意：s1应有足够空间
strncat(s1,s2,n)	将s2的前n个字符连接到s1后面	
strcpy(s1,s2)	将s2的内容复制给s1并替换s1原有的内容	字符串函数
strncpy(s1,s2,n)	将s2的前n个字符复制给s1并替换s1原有的内容	

### 1. 求英文句子中的最长单词

#### 题目描述

一个英文句子（长度不超过255），只含有字母和空格，输出最长的一个单词。如有多个长度相同的单词，则输出最前面的一个。

#### 输入

一个字符串。

#### 输出

一个字符串。

#### 样例输入

```
in which four coins
```

#### 样例输出

```
which
```

### 思路：

打擂法，先用一个数组用来临时存放读出来的单词，另一个数组作为结果，如果新读出来的单词比现有的长，那就换换

### 参考代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char s[300],w[100],r[100]={'\0'};
    int k=0;//标记位
    gets(s);
    for(int i=0;i<strlen(s);i++){
        //不是空格的时候用w存储字符组成单词
        if(s[i]!=' '){
            w[k]=s[i];
            k++;
            //如果下一个也是空格或者没了就不再存字符
            if(s[i+1]==' '||s[i+1]=='\0'){
                w[k]='\0';
                //然后和r对比长度
                if(strlen(w)>strlen(r)){
                    strcpy(r,w);
                }
                k=0;
            }
        }
    }
    cout<<r;
    return 0;
}
```

## 2. 我是第几个单词

### 题目描述

输入一个英文句子，例如：“This is a Book.”，可以看到句子是以“.”来作为结束符号的，并且单词之间以一个空格来分隔。接着再输入一个单词A，请找出首次在句子中出现的与A相同的单词，是句子中的第几个单词，若不存在，则输出该句子中单词字符的总个数。例如对上句子而言，若输入单词“is”，则应输出：2 若输入单词“isa”，则应输出：11

### 输入

第一行为以‘.’结束的一个词组（仅由若干个单词组成，单词间由一空格隔开，除单词和最后的“.”以外，不含其它字符）

第二行是一个单词（不含空格）

### 输出

一个整数

### 样例输入

```
This is a Book.  
Book
```

### 样例输出

### 思路：

与上一题目类似，挨个取出单词与要查找的单词进行匹配，如果成功则输出位置并break，如果读到'\0'还没有找到，则输出字符数。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char s[300],w[100]={'\0'},t[100];
    int c=0,k=0,f=0;
    gets(s);
    gets(t);
    for(int i=0;i<strlen(s);i++){
        if(s[i]!=' '){
            w[k]=s[i];
            k++;
            c++;
            if(s[i+1]==' '||s[i+1]=='\0'||s[i+1]=='.'){
                w[k]='\0';
                f++;
                if(!strcmp(w,t)){
                    cout<<f;
                    break;
                }
                if(s[i+1]=='\0'){
                    cout<<c;
                }
                k=0;
            }
        }
    }
    return 0;
}
```

### 3. 简单a+b

#### 题目描述

张晓菲同学做了简单的ab求和的问题。但是，如果要求输入的情况不是a和b，而是整个加法表达式呢？请想办法，计算加法表达式的结果。

#### 输入

输入一个加法表达式，如 $1+2=$ ，或者 $23+58=$ 。（注意：做加法的2个整数都在0~999999999的范围内）

#### 输出

计算出输入表达式的正确结果

#### 样例输入

1+2=

#### 样例输出

## 思路

先剥离两个“加数”，然后再转化为真正的数字

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char s[30],s1[15],s2[15];
    int f,i,k=0,n1=0,n2=0;
    gets(s);
    for(i=0;i<strlen(s);i++){
        if(s[i]=='+' ){
            f=i;
            break;
        }
    }
    for(i=0;i<f;i++){
        if(s[i]!=' '){
            s1[k]=s[i];
            k++;
        }
    }
    s1[k]='\0';

    k=0;
    for(i=f+1;i<strlen(s)-1;i++){
        if(s[i]!=' '){
            s2[k]=s[i];
            k++;
        }
    }
    s2[k]='\0';
    k=0;
    while(s1[k]!='\0'){
        n1=n1*10+(s1[k]-48);
        k++;
    }
    k=0;
    while(s2[k]!='\0'){
        n2=n2*10+(s2[k]-48);
        k++;
    }
    cout<<n1+n2;
    return 0;
}
```

## 3.课后作业

### 1. 统计单词个数

#### 题目描述

输入一行字符串，包含若干个单词，约定相邻的两个单词用空格隔开（一个或多个空格），编程统计单词的个数

## 输入

一行空格隔开的若干个单词。

## 输出

单词个数

### 样例输入

```
Hello    world
```

### 样例输出

```
2
```

## 2. 找最长单词

### 题目描述

编写程序，根据给出的一个结束于'.'的字符字串，找出其中最长的含有字母'a'的子串。

## 输入

一行，为一个字符字串，结束于句点'.'。字串中的子串由一个或几个空格隔开。

## 输出

一行。显示找出的最长的含有字母'a'的子串。如果有多个这样的子串，只显示其中的第一个；若没有含字母'a'的子串，则显示'NO'。

### 样例输入

```
Her name is Lilan and she is a happy student.
```

### 样例输出

```
Lilan
```

## 3. 简单a\*b

### 题目描述

按照 $a*b$ =的格式输入算式，通过计算输出 $a*b$ 的结果。

## 输入

输入中包括一个表达式，如： $a*b$ = a和b都是int类型的正整数。

## 输出

结果只有一个正整数，整数在long long范围内。

### 样例输入

```
100*200=
```

### 样例输出

```
20000
```

#### 4. 合法的变量名？

##### 题目描述

James在一节C++课程上，准备为自己的程序定义变量名称。老师告诉James，一个合法的变量名应该满足如下的三个条件：

- 1、只能由字母（大写或者小写）、数字及下划线（\_）组成
- 2、不能以数字开头
- 3、不能是C++中有特殊含义的单词，由于James是C++的初学者，只学过int、double、cout、cin这四个有特殊含义的单词，因此James只要避开这几个单词就可以。

请你编程帮助James判断他定义的变量名是否合法。

下表中列举了一些合法的变量名和非法的变量名的案例供你参考。

合法变量名案例	非法变量名案例
x	sum x: 有空格（只能是字母数字或下划线）
sum	1x: 以数字开头
sum_x	int: 有特殊含义
sum2	a#: 有特殊字符“#”（只能是字母、数字、下划线）
_sum	
INT注意：这是合法的，因为在C++中是区分大小写的，int是表示整数类型，但INT不是）；	

##### 输入

一行，包含一个字符串，是James为变量起的名字，且长度不大于20。

##### 输出

一行，如果是合法的C++变量名，则输出yes，否则输出no。

##### 样例输入

```
sum
```

##### 样例输出

```
yes
```

#### 5. 重新排列

##### 题目描述

现在有一个20位以内的自然数，你可以将组成这个数的各位数字重新排列，得到一个数值为最小的新数，但新数的位数保持不变。请编程打印出重新排列后的新数。（如：231重新排序后位数不变的最小数是123，而23105重新排序后位数不变的最小数是10235。）

##### 输入

一个整数（位数  $\leq 20$ ）

#### 输出

重新排列后最小的新数

#### 样例输入

```
382
```

#### 样例输出

```
238
```

## 6. 趣味填空

#### 题目描述

小华的寒假作业上，有这样一个趣味填空题：给出用等号连接的两个整数，如“1234 = 127”。当然，现在这个等号是不成立的。题目让你在左边的整数中间某个位置插入一个加号，看有没有可能让等号成立。以上面的式子为例，如果写成 $123+4=127$ ，这就可以了。请你编写一个程序来解决它。

#### 输入

只有那个不相等的式子。已知，等号两边的整数都不会超过2000000000。

#### 输出

如果存在这样的方案，请输出那个正确的式子。如果不存在解决方案，请输出“Impossible!”（引号中的部分）。

#### 样例输入

```
1234=127
```

#### 样例输出

```
123+4=127
```

# 第二十一章 二维数组

## 1. 什么是二维数组

### 1. 定义：什么是二维数组

二维数组，即存储数组的数组（数组的元素是数组）

数组的作用：定义一个变量存储多个值，作为同一类型元素存储的“容器”

**例子：**准备一个数组存储5年级2班所有同学的语文成绩

89	88	88	92	91	85	96	96	94
0	1	2	3	4	5	6	7	8

数组的每一格都代表一个同学的成绩，我们可以通过下脚标来访问数组元素，但如果需要存储一个班所有同学的语文、数学、英语成绩，一个数组还可以完成任务吗？

**例子：**如何存储一个班的所有同学的多门成绩？

i=0	96	88	86	87	92	89	95	94	95
i=1	95	95	98	93	85	91	94	98	91
i=2	94	91	86	90	89	93	87	86	87
	j=0	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8

注意：

1. 二维数组可以理解为数组中还有数组
2. 二维数组的访问方式：先描述行下脚标，再描述列下脚标
2. 二维数组的定义方式
  1. 定义不赋值，元素的值未知

```
int a[3][4]; // 定义一个3*4的二维数组，但没有为数组的元素赋值，初始值未知，原理同变量
```

```
/*运行结果如下： */
a[0][0]1906224
a[0][1]0
a[0][2]1
a[0][3]0
a[1][0]-1
a[1][1]-1
a[1][2]4253621
a[1][3]0
a[2][0]1
a[2][1]0
a[2][2]4254553
a[2][3]0
```

补：遍历二维数组的方式：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a[3][4], i, j;
    // 外循环就是循环第一层数组的元素，可以理解为定义了一个名叫a的数组，里面有三个元素，这些元素是数组，而每一个作为元素的数组里都有四个元素
    for(i=0; i<3; i++){
        // 第二层就是遍历出作为元素的数组里的元素
        for(j=0; j<4; j++){
            cout << "a[" << i << "] " << "[" << j << "] " << a[i][j] << endl;
        }
    }
    return 0;
}
```

2. 定义的同时赋值

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    // 为了更直观地理解二维数组的构成，可以写成如下的形式（非必须）
    int i, j, a[3][4] = {
```

```

{1,2,3,4},
{5,6,7,8},
{9,10,11,12}
};

for(i=0;i<3;i++){
    for(j=0;j<4;j++){
        cout<<"a["<<i<<"]"<<"["<<j<<"]"<<a[i][j]<<endl;
    }
}
return 0;
}

```

3. 定义时赋值，但数据不足数组实际长度，顺序拿到值后剩下的值默认为0

```
int a[3][4]={1,2,3,4};
```

运行结果如下：

```

a[0][0]1
a[0][1]2
a[0][2]3
a[0][3]4
a[1][0]0
a[1][1]0
a[1][2]0
a[1][3]0
a[2][0]0
a[2][1]0
a[2][2]0
a[2][3]0

```

## 2.课堂练习

### 1. 小黄摘苹果

#### 题目描述

小黄有一天走到了一片苹果林，里面每棵树上都结有不同数目的苹果，小黄身上只能拿同一棵树上的苹果，他每到一棵果树前都会把自己身上的苹果扔掉并摘下他所在树上的苹果并带走（假设小黄会走过每一棵苹果树），问在小黄摘苹果的整个过程中，他身上携带的最多苹果数与最小苹果数的差是多少？

#### 输入

m, n (即苹果林中有果树的行数和列数, 0<n,m<=10) m行n列数据 (即每棵树上的苹果数)

#### 输出

1个数字 (小黄摘苹果的整个过程中，他身上携带的最多苹果数与最小苹果数的差)

#### 样例输入

```
4 3
2 6 5
1 3 7
5 3 5
1 7 12
```

### 样例输出

```
11
```

### 思路：

求出二维数组中的最大值与最小值即可

## 2. 求各个科目成绩的平均分

### 题目描述

请从键盘读入一个整数n (n<=100)，代表一个班级同学的人数，然后读入n个人的语文、数学、英语成绩；请求出这n个人的语文、数学、英语三科成绩的平均分分别是多少，结果保留1位小数。

### 输入

第一行：一个整数n，代表班级的人数！

第2行~第n+1行，输入n个同学的语文、数学、英语成绩，每行输入一个同学的成绩，成绩用空格隔开！

### 输出

输出语文、数学、英语三科的平均成绩，分别用空格隔开，平均成绩保留1位小数！

### 样例输入

```
2
100 99 98
99 98 97
```

### 样例输出

```
99.5 98.5 97.5
```

思路：对每列成绩求和并计算平均值即可

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,a[101][3],s1=0,s2=0,s3=0,m;
    cin>>m;
    for(i=0;i<m;i++){
        for(j=0;j<3;j++){
            cin>>a[i][j];
        }
        s1+=a[i][0];
        s2+=a[i][1];
        s3+=a[i][2];
    }
    cout<<fixed<<setprecision(1)<<s1*1.0/m<<" ";
    cout<<fixed<<setprecision(1)<<s2*1.0/m<<" ";
}
```

```
    cout<<fixed<<setprecision(1)<<s3*1.0/m<<" ";
    return 0;
}
```

### 3. 输出杨辉三角的前N行

#### 题目描述

输出杨辉三角的前N行(N<10)。

#### 输入

输入只有一行，包括1个整数N。(N<10)

#### 输出

输出只有N行.

#### 样例输入

```
5
```

#### 样例输出

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

### 4. 图像相似度

#### 题目描述

给出两幅相同大小的黑白图像（用0-1矩阵，0代表白色，1代表黑色）表示，求它们的相似度。

说明：若两幅图像在相同位置上的像素点颜色的值相同，则称它们在该位置具有相同的像素点。两幅图像的相似度定义为相同像素点数占总像素点数的百分比。

比如：输入

```
2 2
1 0
0 1
1 1
1 1
```

数据解释：第一行的2 2表示图像的尺寸是2行，每行2个整数；接下来的两行数据1 0和0 1表示第一幅图片的数值，再接下来两行数据1 1和1 1表示第二幅图片的数值；从数据上可以看出，两幅图片有2个数是相等的，因此两幅图片的相似度为50%，实际输出不需要输出百分号，结果保留2位小数，因此实际输出50.00。

#### 输入

第一行包含两个整数n和m，表示图像的行数和列数，中间用单个空格隔开。 $1 \leq n \leq 100$ ,  $1 \leq m \leq 100$ 。

之后n行，每行m个整数0或1，表示第一幅黑白图像上各像素点的颜色。相邻两个数之间用单个空格隔开。

之后n行，每行m个整数0或1，表示第二幅黑白图像上各像素点的颜色。相邻两个数之间用单个空格隔开。

#### 输出

一个小数，表示相似度（以百分比的形式给出，但百分号不需要显式），精确到小数点后两位。

### 样例输入

```
3 3
1 0 1
0 0 1
1 1 0
1 1 0
0 0 1
0 0 1
```

### 样例输出

```
44.44
```

**思路：**统计出相同的点的数量，然后除以总的像素点即可

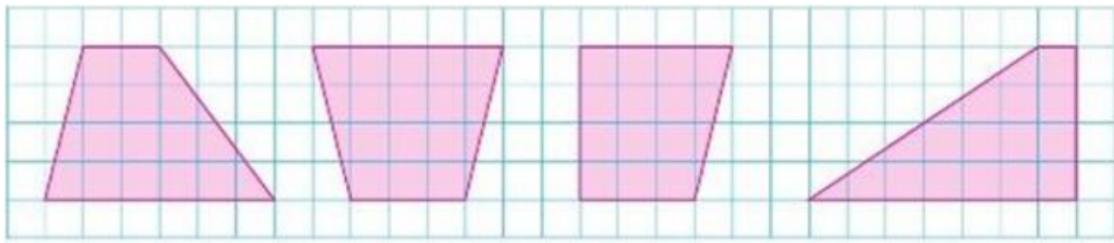
```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,a[110][110],b[110][110],m,n,c=0;
    cin>>m>>n;
    for(i=0;i<m;i++){
        for(j=0;j<n;j++){
            cin>>a[i][j];
        }
    }
    for(i=0;i<m;i++){
        for(j=0;j<n;j++){
            cin>>b[i][j];
        }
    }
    for(i=0;i<m;i++){
        for(j=0;j<n;j++){
            if(a[i][j]==b[i][j]){
                c++;
            }
        }
    }
    cout<<fixed<<setprecision(2)<<c*100.0/(m*n);
    return 0;
}
```

## 3.课后作业

### 1. 求最大梯形的面积

#### 题目描述

从键盘读入n(3<=n<=100)个梯形的上底、下底和高，请问这n个梯形中，最大面积的梯形的面积是多少？（梯形面积的求解公式为  $S = (a + b) * h / 2$ ，也就是(上底 + 下底) \* 高 / 2）



### 输入

第1行为1个整数n，接下来n行每行3个整数分别代表梯形的上底、下底和高。

### 输出

最大面积梯形的面积（结果保留1位小数）

### 样例输入

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

### 样例输出

```
17.5
```

## 2. 靶心数

### 题目描述

James同学发现了在二维数组中有这样一类数，这个数正好比自己上下左右四个方向的数都大（由于需要比四个方向的数都大，因此这个数不可能在第一行、最后一行、第一列、最后一列），James把它们称为靶心数，请你编程求出一个二维数组的靶心数有哪些，输出他们。

### 输入

第一行是两个整数n和m（n和m都是4~100之间的整数），代表接下来的二维数组有n行m列。  
接下来n行，每行有m个整数。

### 输出

请按照输入的顺序输出满足条件的靶心数，每行1个。

### 样例输入

```
4 4
1 2 3 4
5 6 5 8
9 1 11 10
13 4 5 16
```

### 样例输出

```
6
11
```

## 3. 找回文数

### 题目描述

James同学发现了在二维数组中有一些回文数，请编程找出这些回文数，并按照输入的顺序输出。  
(回文数指的是这个数正过来读和反过来读是同一个数的数，比如1、8、99、252、1221等)。

### 输入

第一行是两个整数n和m (n和m都是4~100之间的整数)，代表接下来的二维数组有n行m列。  
接下来n行，每行有m个整数，这些整数都是1~9999之间的整数。

### 输出

按照输入的顺序输出满足条件的回文数，每行1个。

### 样例输入

```
3 3
1 22 98
34 121 110
100 210 323
```

### 样例输出

```
1
22
121
323
```

## 4. 奇偶统计

### 题目描述

在一个n行m列的二维数组中，有若干奇数和偶数，请编程统计出这个二维数组中，奇数和偶数分别有多少个？

### 输入

第一行是两个整数n和m (n和m都是4~100之间的整数)，代表接下来的二维数组有n行m列。  
接下来n行，每行有m个整数。 (这些整数都是0~9999之间的整数)

### 输出

两个整数用空格隔开，分别代表二维数组中奇数、偶数的个数

### 样例输入

```
2 2
2 3
4 6
```

### 样例输出

```
1 3
```

## 5. 石头剪刀布

### 题目描述

石头剪刀布是常见的猜拳游戏。石头胜剪刀，剪刀胜布，布胜石头。如果两个人出拳一样，则不分胜负。一天，小a和小b正好在玩石头剪刀布，假设1代表石头，2代表剪刀，3代表布。小a和小b一共玩了n轮，请问最后的比赛结果是小a赢了还是小b赢了，还是平局？

注意：最终输赢按照小a和小b赢的总次数计算。例如：共比赛7局，小a赢了4局，小b赢了3局，那么输出“a win”。

### 输入

第一行，是一个整数n (n<=100)

接下来n行，每行有2个数，分别代表每轮比赛中小a和小b的出拳。

### 输出

如果小a赢了，输出字符串“a win”，如果小b赢了，输出字符串“b win”，如果平局则输出字符串“tie”。（请注意：输出的字符串全部是小写）

### 样例输入

```
3  
1 2  
2 3  
1 1
```

### 样例输出

```
a win
```

## 4.二维数组矩阵类问题解题思路

1. 观察二维数组需要赋值几次；
2. 观察二维数组，第i次赋值几个数，这几个数的下脚标规律是什么；
3. 观察赋值的规律

## 5.二维数组矩阵类问题课堂练习

### 1. 对角线I

#### 题目描述

输入整数N，输出相应方阵。

#### 输入

一个整数N。 ( 0 < n < 10 )

#### 输出

一个方阵，每个数字的场宽为3。

### 样例输入

```
5
```

### 样例输出

```
1  0  0  0  0  
0  1  0  0  0  
0  0  1  0  0  
0  0  0  1  0  
0  0  0  0  1
```

**补充：**数字的场宽是指输出时所占的空间大小。如本题所示，场宽为3就是在数字前有2个空格，加上本身数字就有了3个位置。如果数字位数大于3，如5432，场宽为3时不会减少位数，而是正常输出，只有当本身不足3个位置时才会在数字前补充空格。设置场宽的方法是setw()。

**示例代码：**

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,n,a[11][11]={0};
    cin>>n;
    for(i=0;i<n;i++){
        a[i][i]=1;
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}
```

## 2. 对角线II

**题目描述**

输入整数N，输出相应方阵。

**输入**

一个整数N。 ( 0 < n < 10 )

**输出**

一个方阵，每个数字的场宽为3。

**样例输入**

```
5
```

**样例输出**

```
0  0  0  0  1
0  0  0  1  0
0  0  1  0  0
0  1  0  0  0
1  0  0  0  0
```

**示例代码：**

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,n,a[11][11]={0};
    cin>>n;
    for(i=0;i<n;i++){
        a[i][n-1-i]=1;
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<setw(3)<<a[i][j];
```

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

### 3. 数字走向I

#### 题目描述

输入整数N，输出相应方阵。

#### 输入

一个整数N。 ( 0 < n < 10 )

#### 输出

一个方阵，每个数字的场宽为3。

#### 样例输入

```
5
```

#### 样例输出

```
1  2   3   4   5
6   7   8   9  10
11  12  13  14  15
16  17  18  19  20
21  22  23  24  25
```

#### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,n,a[11][11]={0},c=1;
    cin>>n;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            a[i][j]=c;
            c++;
        }
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}
```

### 4. 数字走向II

#### 题目描述

输入整数N，输出相应方阵。

## 输入

一个整数N。 ( 0 < n < 10 )

## 输出

一个方阵，每个数字的场宽为3。

### 样例输入

```
5
```

### 样例输出

```
21 22 23 24 25
16 17 18 19 20
11 12 13 14 15
 6  7  8  9 10
 1  2  3  4  5
```

### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,n,a[11][11]={0},c=1;
    cin>>n;
    for(i=n-1;i>=0;i--){
        for(j=0;j<n;j++){
            a[i][j]=c;
            c++;
        }
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}
```

## 5. 数字走向III

### 题目描述

输入整数N，输出相应方阵。

## 输入

一个整数N。 ( 0 < n < 10 )

## 输出

一个方阵，每个数字的场宽为3。

### 样例输入

**样例输出**

```
1 6 11 16 21
2 7 12 17 22
3 8 13 18 23
4 9 14 19 24
5 10 15 20 25
```

**示例代码**

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,n,a[11][11]={0},c=1;
    cin>>n;
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            a[j][i]=c;
            c++;
        }
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}
```

**6. 数字走向IV****题目描述**

输入整数N，输出相应方阵。

**输入**

一个整数N。 ( 0 < n < 10 )

**输出**

一个方阵，每个数字的场宽为3。

**样例输入****样例输出**

```
21 16 11 6 1
22 17 12 7 2
23 18 13 8 3
24 19 14 9 4
25 20 15 10 5
```

### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,n,a[11][11]={0},c=1;
    cin>>n;
    for(i=n-1;i>=0;i--){
        for(j=0;j<n;j++){
            a[j][i]=c;
            c++;
        }
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}
```

## 7. 斜角II

### 题目描述

输入整数N，输出相应方阵。

### 输入

一个整数N。 ( 0 < n < 10 )

### 输出

一个方阵，每个数字的场宽为3。

### 样例输入

```
5
```

### 样例输出

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

### 示例代码

```
#include<bits/stdc++.h>
```

```

using namespace std;
int main(){
    int i,j,n,a[11][11]={0},c=1;
    cin>>n;
    for(i=0;i<n;i++){
        for(j=0;j<=i;j++){
            a[i-j][j]=c;
            a[n-1-j][j+n-1-i]=c;
        }
        c++;
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}

```

## 8. 斜角I

### 题目描述

输入整数N，输出相应方阵。

### 输入

一个整数N。 ( 0 < n < 10 )

### 输出

一个方阵，每个数字的场宽为3。

### 样例输入

5

### 样例输出

1	2	3	4	5
2	3	4	5	6
3	4	5	6	7
4	5	6	7	8
5	6	7	8	9

### 示例代码

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,n,a[11][11]={0};
    cin>>n;
    for(i=0;i<n;i++){
        for(j=0;j<=i;j++){
            a[i-j][j]=i+1;
            a[n-1-j][j+n-1-i]=2*n-1-i;
        }
    }
}

```

```
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}
```

## 9. 拐角I

### 题目描述

输入整数N，输出相应方阵。

### 输入

一个整数N。 ( 0 < n < 10 )

### 输出

一个方阵，每个数字的场宽为3。

### 样例输入

```
5
```

### 样例输出

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

### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,n,a[11][11]={0};
    cin>>n;
    for(i=0;i<n;i++){
        for(j=0;j<n-i;j++){
            a[i][j+i]=i+1;
            a[j+i][i]=i+1;
        }
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}
```

## 10. 有趣的数字图形II

### 题目描述

输入一个整数n (n≤12) , 打印出如下要求的方阵：左上到右下对角线上的数与行数相同，右上半部分区域中每个元素等于左边的和下面的元素之和。每个元素场宽为5。左下半个区域为空。

### 输入

一个整数n (n≤12)

### 输出

n\*n的方阵 (场宽为5) 。

### 样例输入

4

### 样例输出

1	3	8	20
2	5	12	
3	7		
4			

### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,j,n,a[11][11]={0};
    cin>>n;
    for(i=0;i<n;i++){
        for(j=0;j<n-i;j++){
            if(i==0){
                a[j][j+i]=j+1;
            }else{
                a[j][j+i]=a[j][j+i-1]+a[j+1][j+i];
            }
        }
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            if(a[i][j]!=0){
                cout<<setw(5)<<a[i][j];
            }else{
                cout<<setw(5)<<' ';
            }
        }
        cout<<endl;
    }
    return 0;
}
```

## 6.二维数组矩阵类课后作业

### 1. 数字走向V

#### 题目描述

输入整数N，输出相应方阵。

#### 输入

一个整数N。 ( 0 < n < 10 )

#### 输出

一个方阵，每个数字的场宽为3。

#### 样例输入

```
5
```

#### 样例输出

```
25 24 23 22 21
20 19 18 17 16
15 14 13 12 11
10  9   8   7   6
 5   4   3   2   1
```

### 2. 数字走向VI

#### 题目描述

输入整数N，输出相应方阵。

#### 输入

一个整数N。 ( 0 < n < 10 )

#### 输出

一个方阵，每个数字的场宽为3。

#### 样例输入

```
5
```

#### 样例输出

```
 5   4   3   2   1
10   9   8   7   6
15 14 13 12 11
20 19 18 17 16
25 24 23 22 21
```

### 3. 斜角I

#### 题目描述

输入整数N，输出相应方阵。

#### 输入

一个整数N。 ( 0 < n < 10 )

## **输出**

一个方阵，每个数字的场宽为3。

## **样例输入**

```
5
```

## **样例输出**

```
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
```

## **4. 斜角IV**

### **题目描述**

输入整数N，输出相应方阵。

### **输入**

一个整数N。 ( 0 < n < 10 )

### **输出**

一个方阵，每个数字的场宽为3。

## **样例输入**

```
5
```

## **样例输出**

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

## **5. 斜角III**

### **题目描述**

输入整数N，输出相应方阵。

### **输入**

一个整数N。 ( 0 < n < 10 )

### **输出**

一个方阵，每个字母的场宽为3。

## **样例输入**

```
5
```

## **样例输出**

A	B	C	D	E
B	C	D	E	A
C	D	E	A	B
D	E	A	B	C
E	A	B	C	D

## 6. 拐角II

### 题目描述

输入整数N，输出相应方阵。

### 输入

一个整数N。 ( 0 < n < 10 )

### 输出

一个方阵，每个数字的场宽为3。

### 样例输入

5

### 样例输出

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

## 7. 拐角III

### 题目描述

输入整数N，输出相应方阵。

### 输入

一个整数N。 ( 0 < n < 10 )

### 输出

一个方阵，每个数字的场宽为3。

### 样例输入

5

### 样例输出

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

## 8. 拐角IV

### 题目描述

输入整数N，输出相应方阵。

#### 输入

一个整数N。 ( 0 < n < 10 )

#### 输出

一个方阵，每个数字的场宽为3。

#### 样例输入

```
5
```

#### 样例输出

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

### 9. 有趣的数字图形I

#### 题目描述

读入一个整数n (n<100) ,输出一个n\*n的方阵。对角线是1，对角线向右上逐渐递增，左下为空格，每个数场宽为5。

#### 输入

一个整数n

#### 输出

n\*n的方阵

#### 样例输入

```
4
```

#### 样例输出

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

### 10. 有趣的数字图形III

#### 题目描述

读入一个整数n (n<100) ,输出一个n\*n的方阵。对角线是1，对角线向右下逐渐递增，左上为空格，每个数场宽为5。

#### 输入

一个整数n

#### 输出

n\*n的方阵

#### 样例输入

**样例输出**

		1	
	1	2	
1	2	3	
1	2	3	4

**11. 有趣的数字图形IV****题目描述**

输入一个整数n ( $n \leq 12$ ) , 打印出如下要求的方阵：除掉右上到左下对角线上的数外的右下半个区域中每个元素等于左边的和上面的元素之和。每个元素场宽为5。左上半个区域为空。

**输入**

一个整数n ( $n \leq 12$ )

**输出**

$n \times n$ 的方阵 (场宽为5) 。

**样例输入**

4

**样例输出**

		4	
	3	7	
2	5	12	
1	3	8	20

**12. 有趣的数字图形****题目描述**

输入一个整数n ( $n < 10$ ) , 输出n行n列的如下图所示的数字图形。

**输入**

一个整数n。 ( $1 < n < 10$ )

**输出**

一个方阵，每个数字的场宽为5。

**样例输入**

4

**样例输出**

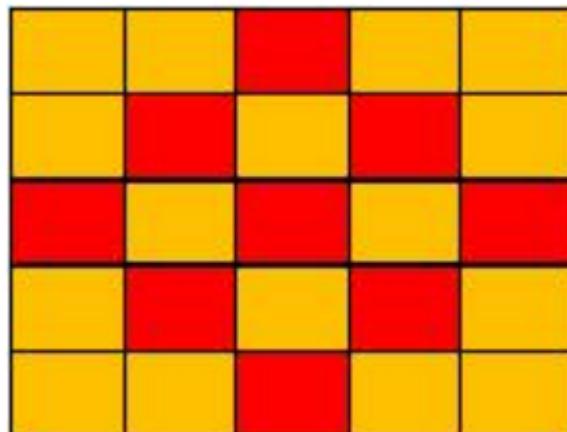
1	3	8	20
3	2	5	12
8	5	3	7
20	12	7	4

### 13. 鲜花方阵

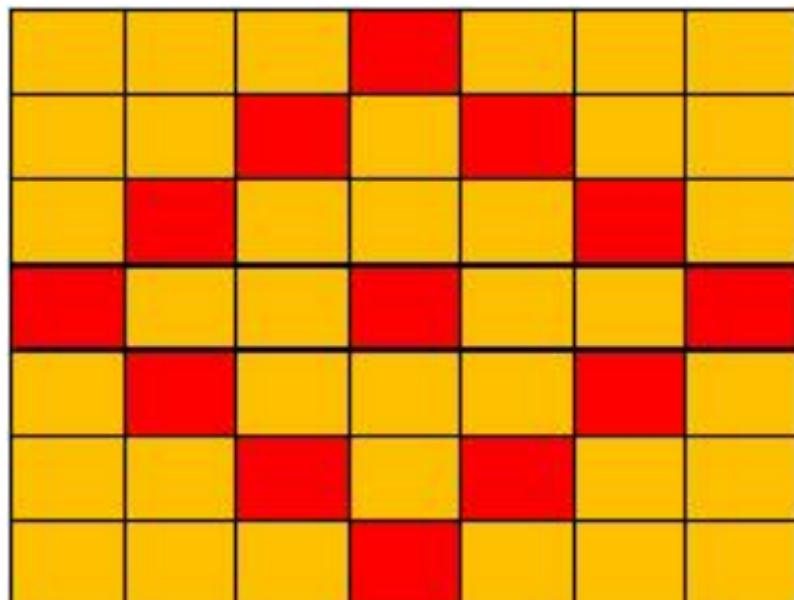
#### 题目描述

光明小学艺术节快要来了，老师要求同学们布置一个 $n * n$ 的花盆方阵（ $n$ 是奇数， $n \leq 9$ ）；

如果 $n=5$ ，那么方阵的形状如下图所示：



如果 $n=7$ ，那么方阵的形状如下图所示：



请读入一个整数 $n$ （奇数），输出如图所示的方阵，为了方便输出，用1表示黄色的花盆，0表示红色花盆，输出时数字场宽设置为3。

例如： $n=5$ ，那么实际要输出的方阵的结果如下

1	1	0	1	1
1	0	1	0	1
0	1	0	1	0
1	0	1	0	1
1	1	0	1	1

$n=7$ , 那么实际要输出的方阵的结果如下

1	1	1	0	1	1	1
1	1	0	1	0	1	1
1	0	1	1	1	0	1
0	1	1	0	1	1	0
1	0	1	1	1	0	1
1	1	0	1	0	1	1
1	1	1	0	1	1	1

### 输入

一个整数n

### 输出

如图所示的图形

### 样例输入

```
5
```

### 样例输出

```
1 1 0 1 1
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
1 1 0 1 1
```

# 码蜂C++程序设计--算法篇

## 第二十二章 string

### 1.string基础知识

#### 1. string 是什么?

string 是C++的STL (Standard Template Library 即标准模板库) 提供的字符串类, 用于处理字符串相关的问题;

#### 2. string 与 char[]的异同

1. 字符数组本质上属于数组, 长度是固定的, string可以理解为长度无限的字符串
2. 字符数组的系统定义的函数较少, 所以操作不方便, 而string集成了大量的系统函数, 操作方便;
3. 字符数组由于本质是数组, 因此无法进行比较运算和+运算, string可以

#### 3. string的特点总结:

1. 长度可以理解为无限制

2. string可以做加法，可以比较大小
3. string有很多系统自带函数
4. 为方便理解，可将string理解为特殊的char数组

## 2.string 的读入与遍历

1. getline(cin,s): 读入一个字符串（直到换行），可以含空格；
2. cin: 读入一个字符串，不可含空格；
3. s.size(): 获取字符串s的长度
4. s[下脚标i]: 获取字符串某个下标对应的字符
5. string 的加法运算，见课堂例题：[打印小写字母表](#)
6. 示例：

### 1. string的读入与遍历基础

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
// cin>>s;//不可包含空格
cout<<s.size()<<endl;
getline(cin,s);//可包含空格
cout<<s<<endl;
//遍历每个字符
int i;//i用作下脚标
for(i=0;i<s.size();i++){
    cout<<s[i]<<endl;
}
string s2="coderbee.cn";//string类型的数据可以直接赋值，无需循环
cout<<s2;
return 0;
}
```

### 2. string 的加法和比较运算

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
//在字符串上叠加字符串
s=s+"Hello,";
s+="coderbee";
cout<<s<<endl;
//在字符串上叠加字符
s+='.';
s+='c';
s+='n';
cout<<s<<endl;
//注：长度为0的空字符串上是无法进行下脚标访问的
s="python";
//修改字符串中的字符
```

```
s[0]='P';
s[1]='Y';
cout<<s;
return 0;
}
```

### 3.课堂练习

#### 1. 打印小写字母表

##### 题目描述

把英文字母表的小写字母按顺序和倒序打印出来。(每行13个)

##### 输入

无

##### 输出

输出四行

##### 样例输出

```
abcdefghijklm
nopqrstuvwxyz
zyxwvutsrqpon
mlkjihgfedcba
```

##### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    char c='a';
    int i=0;
    for(c;c<='z';c++){
        s+=c;
        i++;
        if(i==13||i==26){
            s+="\n";
        }
    }
    i=0;
    for(c='z';c>='a';c--){
        s+=c;
        i++;
        if(i==13||i==26){
            s+="\n";
        }
    }
    cout<<s;
    return 0;
}
```

## 2. 时间的差

### 题目描述

看到两个标准格式的时间，有小时，有分钟，有秒，格式如：h: m: s 即 时:分:秒 你想知道，这两个时间之间相差多少吗？

### 输入

输入包括两行，两行均为一个“时:分:秒”格式的时间。且本题保证第一个时间一定大于第二个时间！

### 输出

输出就是输入的两个时间之间的秒数差。

### 样例输入

```
01:10:10  
00:30:30
```

### 样例输出

```
2380
```

### 示例代码

```
#include<bits/stdc++.h>  
using namespace std;  
//定义一个函数，计算出每个时间到00:00:00时候的差值，然后相减即可  
int seconds(string str){  
    int h = (str[0] - '0')*10+str[1] - '0';  
    int m = (str[3] - '0')*10+str[4] - '0';  
    int s = (str[6] - '0')*10+str[7] - '0';  
    return h*60*60+m*60+s;  
}  
int main(){  
    string s1,s2;  
    getline(cin,s1);  
    getline(cin,s2);  
    cout<<seconds(s1)-seconds(s2);  
    return 0;  
}
```

## 3. 数字和

### 题目描述

输入一个很大的数，求各位上的数字和。

### 输入

一个很大的整数（不超过200位）

### 输出

一个整数

### 样例输入

**样例输出**

6

**示例代码**

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int i,sum=0;
    string s;
    cin>>s;
    for(i=0;i<s.size();i++){
        sum+=s[i]-'0';
    }
    cout<<sum;
    return 0;
}
```

**4. 补充知识****全局变量与局部变量的区别：**

1. 定义的位置不同：全局在所有函数外，局部变量定义在函数内；
2. 生命周期不同：局部变量仅在定义时最近的大括号内有效，全局变量在所有函数中都有效
3. 初值不同：局部变量不赋值，初始值随机，全局int为0，char为'\0'，小数为0.0，bool为false，int a[100]全为0

**常量的用法**

INT\_MAX:表示整数的最大值;

INT\_MIN:表示整数的最小值;

LONG\_LONG\_MAX: long long 的最大值

... ...

**常见的优化方案：**

减少循环，尽量少定义不必要的数据

**2.1 课堂练习****1. 国王的魔镜****题目描述**

国王有一个魔镜，可以把任何接触镜面的东西变成原来的两倍——只是，因为是镜子嘛，增加的那部分是反的。比如一条项链，我们用AB来表示，不同的字母表示不同颜色的珍珠。如果把B端接触镜面的话，魔镜会把这条项链变为ABBA。如果再用一端接触的话，则会变成ABBAABBA（假定国王只用项链的某一段接触魔镜）。给定最终的项链，请编写程序输出国王没使用魔镜之前，最初的项链可能的最小长度。

**输入**

只有一个字符串，由大写英文字母组成，表示最终的项链。

## 输出

只有一个整数，表示国王没使用魔镜前，最初的项链可能的最小长度。

## 样例输入

```
ABBAABBA
```

## 样例输出

```
2
```

## 思路：

与之前在char类型数组的思路类似，判断是否为偶数数位的回文数，如果是，就取一半，只要修改下取一半的方式即可：

```
#include<bits/stdc++.h>
using namespace std;
bool judge(string s,int end){
    for(int i=0;i<end;i++){
        if(s[i]!=s[end-i]){
            return false;
        }
    }
    return true;
}
int main(){
    int end;
    string str;
    cin>>str;
    if(str.size()%2==0){
        end=str.size()-1;
        while(judge(str,end)){
            end/=2;
        }
    }
    cout<<end+1;
    return 0;
}
```

## 2. 简单加密

### 题目描述

Julius Caesar曾经使用过一种很简单的密码。对于明文中的每个字符，将它用它字母表中后5位对应的字符来代替，这样就得到了密文。比如字符A用F来代替。如下是密文和明文中字符的对应关系。

密文： A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

明文： V W X Y Z A B C D E F G H I J K L M N O P Q R S T U

你的任务是对给定的密文进行解密得到明文。

你需要注意的是，密文中出现的字母都是大写字母。密文中也包括非字母的字符，对这些字符不用进行解码。

## 输入

一行，给出密文，密文不为空，而且其中的字符数不超过200。

## 输出

输出一行，即密文对应的明文。

### 样例输入

```
NS BFW, JAJSYX TK NRUTWYFSHJ FWJ YMJ WJXZQY TK YWNANFQ HFZXJX
```

### 样例输出

```
IN WAR, EVENTS OF IMPORTANCE ARE THE RESULT OF TRIVIAL CAUSES
```

### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    getline(cin,s);
    for(int i=0;i<s.size();i++){
        if(s[i]>='A'&&s[i]<='Z'){
            if((s[i]-5)>='A'){
                cout<<(char)(s[i]-5);
            }else{
                cout<<(char)(s[i]-5+26);
            }
        }else{
            cout<<s[i];
        }
    }
    return 0;
}
```

## 3. 查找字典码最小的字符串

### 题目描述

编写程序，针对输入的N个不同的字符串，输出其中字典码最小的字符串。

### 输入

输入第一行给出正整数N；随后N行，每行给出一个长度小于80的非空字符串，其中不会出现换行符，空格，制表符。

### 输出

输出字典码最小的字符串

### 样例输入

```
5
Li
Wang
Zha
Jin
Xian
```

### 样例输出

```
Jin
```

## 示例代码

## 2.2 课后作业

### 1. 字符串对比

#### 题目描述

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100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

包括两行，每行都是一个字符串

#### 输出

仅有一个数字，表明这两个字符串的关系编号

#### 样例输入

```
BEljing  
beIjing
```

#### 样例输出

```
3
```

### 2. 出现次数最多的小写字母

#### 题目描述

输入一个由小写字母组成的字符串（字符数量 $\leq 100$ ），输出出现次数最多的小写字母。

注意：如果有多个小写字母出现的次数一样多，则输出ASCII码值最大的那个字母。

#### 输入

一个字符串

#### 输出

出现次数最多的小写字母

#### 样例输入

```
aaabbbbbbbcdxs
```

#### 样例输出

b

### 3. 判断是否构成回文

#### 题目描述

输入一串字符,字符个数不超过100,且以"."结束。 判断它们是否构成回文。

#### 输入

输入只有一行, 包括一串字符.

#### 输出

输出只有一行.TRUE 或者FALSE

#### 样例输入

12321.

#### 样例输出

TRUE

### 4. 字符串中的空格移位

#### 题目描述

输入一个字符串, 将其中的所有空格都移到最前面, 然后输出。

#### 输入

一个字符串。如: a□b□c (为了能看清空格, □代表一个空格, 但实际测试数据中不用□表示空格)

#### 输出

空格全部移到了串前的字符串。如: □□abc

#### 样例输入

a b c

#### 样例输出

abc

### 5. 删除字符串中间的\*

#### 题目描述

输入一个字符串, 将串前和串后的保留, 而将中间的删除。

#### 输入

一个含\*的字符串。

#### 输出

删除了串中的\*的字符串。

#### 样例输入

\*\*\*ABC123\*\*\*123\*abc\*\*\*\*\*

## 样例输出

```
***ABC123123abc*****
```

## 6. 字符串的反码

### 题目描述

一个二进制数，将其每一位取反，称之为这个数的反码。下面我们定义一个字符的反码。如果这是一个小写字符，则它和字符'a'的距离与它的反码和字符'z'的距离相同；如果是一个大写字符，则它和字符'A'的距离与它的反码和字符'Z'的距离相同；如果不是上面两种情况，它的反码就是它自身。举几个例子，'a'的反码是'z'；'c'的反码是'x'；'W'的反码是'D'；'1'的反码还是'1'；'!'的反码还是'!'。一个字符串的反码定义为其所有字符的反码。我们的任务就是计算给出定字符串的反码。

### 输入

一个长度不超过80个字符的字符串。

### 输出

字符串的反码。

### 样例输入

```
Hello
```

### 样例输出

```
Svoo1
```

## 7. 看完动漫要几天？

### 题目描述

一部完整的动漫共有m分钟 ( $m \leq 1000$ )。为了保护视力，妈妈决定让小明同学每天从xx:xx分看到xx:xx分，请问小明同学需要几天才能看完这部完整的动漫。 (4.1.15)



### 输入

3行，第一行是一个整数m代表动漫的总分钟数；第二行表示每天看动漫的开始时间，第三行表示每天看动漫的结束时间。（确保输入的开始时间 < 结束时间）

### 输出

一个整数，代表小明看完动漫至少需要花的总天数。

### 样例输入

288

17:00

17:32

### 样例输出

9

## 8. 时钟旋转 (2)

### 题目描述

时钟从时间: xx:xx (xx时xx分) , 走到时间: xx:xx (xx时xx分) , 时针共旋转了多少度? (假设第一个时间<=第二个时间, 2个时间都是12小时制, 且两个时间的时间差不超过12小时, 也就是说时针旋转的度数在360度之内, 也就是  $1 \leq \text{时间}1 \leq \text{时间}2 \leq 12$  (4.2.4))



### 输入

2行, 第一行为起始时间 (如: 01:00) , 第二行为结束时间 (如: 01:05)

### 输出

时针旋转的度数 (结果保留1位小数)

### 样例输入

01:00

01:05

### 样例输出

2.5

## 9. 字符串加密?

### 题目描述

在情报传递过程中, 为了防止情报被截获, 往往需要对情报用一定的方式加密, 简单的加密算法虽然不足以完全避免情报被破译, 但仍然能防止情报被轻易的识别。我们给出一种最简的的加密方法, 对给定的一个字符串, 把其中从a-y, A-Y的字母用其后继字母替代, 把z和Z用a和A替代, 其他非字母字符不变, 则可得到一个简单的加密字符串。

### 输入

输入一行, 包含一个字符串, 长度小于80个字符。

### 输出

输出每行字符串的加密字符串。

### 样例输入

```
Hello! How are you!
```

### 样例输出

```
Ifmmp! Ipx bsf zpv!
```

## 3.string的常见函数

### 3.1.1 string的查找与截取

1. find(字串 substr): 查找子字符串第一次出现的下标，没有返回string::npos(值为-1)
2. find(substr,x):在字符串的下标x之后查找字符串substr
3. substr(开始位置i, 子串长度len): 截取子字符串，当len大于字符串长度时，只取剩余的
4. substr(开始位置i): 截取子字符串，从下标为i开始截取到最后

### 3.1.2 用法示例

1. 请求出父字符串s中首次出现子字符串s1的下标，如果s1在s中不存在，则输出 no。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s="Hello,coderbee.cn",s2="Dev C++";
    int p = s.find(s2);
    if(p!=-1){
        cout<<p;
    }else{
        cout<<"no";
    }

    return 0;
}
```

2. 请将一句英文中的第2个单词截取出来，该句子中的单词用空格隔开。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s="Hello coderbee cn";
    int p1 = s.find(" "), p2 = s.find(" ",p1+1);
    cout<<s.substr(p1+1,p2-p1);
    return 0;
}
```

### 3.1.3 课堂练习

1. 删除指定字符

#### 题目描述

请问在一个父字符串s中是否存在子字符串t。如果存在，则输出子字符串t在父字符串中所有的起始位置，如果不存在，则输出-1。

比如：假设父字符串s = "Go Abc good goole!"，子字符串t = "go"，那么输出位置：

8

13

再比如：假设父字符串s = "Go Abc good goole!"，子字符串t = "hi"，那么输出结果： -1。

### 输入格式

第一行输入父字符串的值；

第二行输入子字符串的值；

### 输出格式

输出子字符串在父字符串中所有的位置，如果父字符串中不存在子字符串，请输出-1。

### 输入样例

```
Go Abc good goole!
go
```

### 输出样例

```
8
```

```
13
```

### 解法1:用find求解

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s,t;
    getline(cin,s);
    getline(cin,t);
    int p = s.find(t);
    if(p == -1){
        cout<<-1;
    } else{
        while(p != -1){
            cout<<p+1<<endl;
            p=s.find(t,p+1);
        }
    }
    return 0;
}
```

### 解法2：使用substr求解

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s,t,x;
    getline(cin,s);
    getline(cin,t);
```

```
bool f = false;
int i;
/*
从每个字符开始截取t.size()个字符串出来，判断截取出来的结果是否是要找的t，如果是，就输出i+1
*/
for(i=0;i<s.size();i++){
    x = s.substr(i,t.size());
    if(x == t){
        cout<<i+1<<endl;
        f=true;
    }
}
if(!f){
    cout<<-1;
}
return 0;
}
```

## 2. 字符替换

### 题目描述

将用逗号隔开的两个英语单词交换位置输出。

### 输入

一行以逗号隔开的两个英文单词

### 输出

将两个单词交换后输出的结果

### 样例输入

```
abc,de
```

### 样例输出

```
de,abc
```

### 思路：

将逗号前后的单词截取出来后替换位置即可。

### 参考代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s,s1,s2;
    cin>>s;
    int p = s.find(",");
    s1=s.substr(0,p);
    s2=s.substr(p+1);
    cout<<s2<<" , "<<s1;
    return 0;
}
```

### 3.1.4 课后作业

#### 1. 题目描述

给出两个字符串，将它们进行拼接，拼接过程中每个字符只允许出现一次。

如：两个字符串s1="adeab"，s2="fcadex"，那么连接时s1留下adeb（第2个a出现过了，就不要了），再将s2连接上来形成adebcx，两个字符串中重复的都过滤掉，但剩余的顺序不要调整。

#### 输入

两行，每行一个只包含小写英文字母的字符串。

#### 输出

一行，连接后的字符串。

#### 样例输入

```
abc  
daaeb
```

#### 样例输出

```
abcde
```

#### 2. 题目描述

一个字符串中任意个连续的字符组成的子序列为该字符串的子串。给定子串s1和它的一个字符串s2，求s1在s2中出现的次数。

#### 输入

第一行，表示字符串s1，第二行，表示字符串s2

#### 输出

一个整数，代表s1在s2中出现的次数。

#### 样例输入

```
ab  
abbaabcaabc
```

#### 样例输出

```
3
```

### 3.2.1 string的插入、删除与替换

1. `erase(开始下标i,删除长度len)`: 删除从第i个字符开始的len个字符
2. `erase(开始下标i)`: 删除从第i个字符开始后面的所有字符
3. `insert(插入的下标i,插入字符串s)`: 在字符串下标为i的位置插入一个字符串s
4. `replace(i,len,str)`: 从下标i开始，替换len个字符为str

### 3.2.2 用法示例

erase、insert、replace函数的使用

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s = "This is a book";
//    string s2 = s.substr(5,3);
//    cout<<s<<endl<<s2<<endl;
    s.erase(5,3);
    //this a book
    cout<<s<<endl;
    //插入 is
    s.insert(5,"is ");
    cout<<s<<endl;
    //d:this is a book
    s.replace(10,4,"student");
    cout<<s<<endl;
    return 0;
}
```

### 3.2.3 课堂案例

#### 1. 删除指定字符

##### 题目描述

从键盘输入一个字符串str和一个字符c，删除str中的所有字符c并输出删除后的字符串str。

##### 输入

第一行是一个字符串；（不含空格）

第二行是一个字符。

##### 输出

删除指定字符后的字符串。

##### 样例输入

```
sdf$$$sdf$$
$
```

##### 样例输出

```
sdfsdf
```

##### 示例代码

解决方法1：假删除

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    char c;
    cin>>s>>c;
    for(int i=0;i<s.size();i++){
        if(s[i]!=c){
            cout<<s[i];
        }
    }
    return 0;
}

```

解法2：真删除

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    char c;
    cin>>s>>c;
    int p = s.find(c);
    while(p!=-1){
        s.erase(p,1);
        p = s.find(c);
    }
    cout<<s;
    return 0;
}

```

## 2. 字符替换

### 题目描述

把一个字符串中特定的字符全部用给定的字符替换，得到一个新的字符串。

### 输入

只有一行，由一个字符串和两个字符组成，中间用单个空格隔开。字符串是待替换的字符串，字符串长度小于等于100个字符，且不含空格等空白符；

接下来一个字符为需要被替换的特定字符；

接下来一个字符为用于替换的给定字符。

### 输出

一行，即替换后的字符串。

### 样例输入

```
hello-how-are-you o o
```

### 样例输出

```
hello-how-are-you
```

解法1：

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    char c1,c2;
    cin>>s>>c1>>c2;
    for(int i=0;i<s.size();i++){
        if(s[i]==c1){
            s[i]=c2;
        }
    }
    cout<<s;
    return 0;
}

```

解法2：

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    char c1,c2;
    cin>>s>>c1>>c2;
    int p = s.find(c1);
    while(p!=-1){
        s.replace(p,1,c2);
        p=s.find(c1);
    }
    cout<<s;
    return 0;
}

```

### 3.2.4 课后作业

#### 1. 查找子串并替换

##### 题目描述

对输入的一句子实现查找且置换的功能（找到某个子串并换成另一子串）。

比如：将“abcf abdabc”中的“abc”，替换为“AA”，则替换结果为“AAf abdAA”，而将“abcf abdabc”中的“abc”替换为“abcabc”，则替换结果为“abcabcf abdabcabc”。本题查找子串时注意，大小写完全一致，才能作为子串，比如：在“Abcf abd Abc”中，如果找字符串“abc”是不存在的。

##### 输入

第一行为原来的字符串 第二行为要查找的子串 第三行为要替换成的子串。

##### 输出

只有一行，为替换好的字符串

##### 样例输入

```

abcf abdabc
abc
AA

```

## 样例输出

```
AAf abdAA
```

## 2. 字符串编辑

### 题目描述

从键盘输入一个字符串（长度 $\leq 40$ 个字符），并以字符'.'结束。

例如：'This is a book.' 现对该字符串进行编辑，编辑功能有：

D：删除一个字符，命令的方式为：

D a 其中a为被删除的字符

例如：D s 表示删除字符's'，若字符串中有多个's'，则删除第一次出现的。

如上例中删除的结果为：'Thi is a book.'

I：插入一个字符，命令的格式为：

I a1 a2 其中a1表示插入到指定字符前面，a2表示将要插入的字符。

例如：I s d 表示在指定字符's'的前面插入字符'd'，若原串中有多个's'，则插入在最后一个字符的前面。

如上例中：

原串：'This is a book.'

插入后：'This ids a book.'

R：替换一个字符，命令格式为：

R a1 a2 其中a1为被替换的字符，a2为替换的字符，若在原串中有多个a1则应全部替换。

例如：原串：'This is a book.'

输入命令：R o e

替换后的字符串为：'This is a beek.'

在编辑过程中，若出现被改的字符不存在时，则给出提示信息"Not exist"。

### 输入

每个测试文件只包含一组测试数据，每组输入数据包含两行：

第一行，输入一个字符串，表示原串；

第二行，输入一个字符串，表示命令，数据保证命令不会出错，也就是命令一定严格按照题目要求输入。

### 输出

对于每组输入数据，输出编辑后的字符串，如果被改的字符不存在，则输出"Not exist"（引号不输出）。

## 样例输入

```
This is a book.
```

```
D s
```

## 样例输出

```
Thi is a book.
```

## 4.字符类型判断/转换，数组/string 排序

### 4.1.1 相关函数介绍

1. 字符类型判断函数：(非string函数)

1. isalpha(c): 判断c是否为字母
2. islower(c): 判断c是否为小写
3. isupper(c): 判断c是否为大写
4. isdigit(c): 判断c是否为数字

**说明：返回非0表示真，0为假**

2. 字符转换型函数：(非string函数)

1. tolower(c): 字符转为小写
2. toupper(c): 字符转换为大写

**说明：返回 int**

3. 排序和倒序函数：(非string函数)

1. sort(起始地址,结束地址+1): 数组升序排序
2. reverse(起始地址,结束地址+1): 数组逆序

**注：**头文件 `<algorithm>`

4. 获取头尾指针

1. s.begin(): 获取字符串s的头位置 (指针)
2. s.end(): 获取字符串s的尾 (最后一个字符后面的位置) 位置 (指针)

**注意：**

bool 类型是一个表示真假的类型，true为真，false 为假

bool f = true;//直接赋值

bool f = 3 > 2; //赋值为判断式

bool f = 3 > 2 && 5 == 5;

true 打印结果为1， false打印结果是0

if、while、for的判断，除了写判断式以外还可以是一个整数类型，如果整数为0，则判断为假，非0判断为真。

```
int a= 6;
if(a){
    cout<<"Yes";
}else{
    cout<<"No";
}
```

上述代码输出结果为“Yes”

**因此，不要出现下面永远为真的写法：**

```

int a = 0;
if( a == 5 ){
    cout<<"Yes";
}else{
    cout<<"No";
}

```

## 4.1.2 用法示例

### 示例：is函数的使用

```

char c;
cin>>c;
//包含大写字母或小写字母
//c>='a'&&c<='z' || c>='A'&&c<='Z'
if(isalpha(c)){
    cout<<"是字母";
    if(islower(c)){
        cout<<"小写";
    }else{
        cout<<"大写";
    }
    //c>='0'&&c<='9'
}else if(isdigit(c)){
    cout<<"是数字";
}

```

### 示例：to函数的使用

```

char c;
c='b';
//将ASCII码赋值给字符，获取编码对应的字符，
//直接打印toupper()会获得ASCII码66
c = toupper(c);
cout<<c;

```

### 示例：sort和reverse函数的使用

有数组如下：

10	8	9	10	5	4					
0	1	2	3	4	5	6	7	8	9	
首地址	地址	地址	地址	地址	地址	地址				
a	a+1	a+2	a+3	a+4	a+5	a+n				

```

#include<bits/stdc++.h>
using namespace std;
//辅助降序排序
bool cmp(int a,int b){
    if(a>b){
        return true;
    }else{
        return false;
    }
}

```

```

    }
}

int main(){
    int a[6] = {10,8,9,6,5,4};
    int i,n = 6 ;
    sort(a,a+n); //对所有数组进行升序排序
    //下面对数组第2个元素~倒数第2个元素升序排序
    sort(a+1,a+n-1);
    sort(a+1,a+n-1,cmp);
    //数组逆序函数
    reverse(a,a+n);
    for(i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    return 0;
}

```

**注意：**

1. sort()第三个参数可以默认不写，如果不填sort会默认按数组升序排序，也可以自定义一个排序函数实现倒序；
2. 数组的本质是数组下标为0的元素的地址（首元素地址）；

**示例代码：使用sort()对字符串进行排序**

```

string s = "adsfgdshgfdjhgfjytfewhnbfd";
sort(s.begin(),s.end());
cout<<s;

```

## 4.2 课堂案例

### 1. 统计字符的个数

#### 题目描述

从键盘中任意输入一串字符,直至输入"#"字符代表结束.请编程统计输入的字符中的大写字母,小写字母和数字字符的个数分别是多少?

#### 输入

输入只有一行,包括一串字符.(长度小于20)

#### 输出

输出只有一行 (这意味着末尾有一个回车符号) , 包括3个整数。分别代表大写字符, 小写字符和数字字符的个数。

#### 样例输入

```
daDSALDdcada3240#
```

#### 样例输出

```
5 7 4
```

#### 示例代码

```

#include<bits/stdc++.h>
using namespace std;
int c1,c2,c3,i;
string s;
int main(){
    getline(cin,s);
    for(i=0;i<s.size();i++){
        //如果是大写字母
        if(isupper(s[i])){
            c1++;
        } else if(islower(s[i])){
            c2++;
        }else if(isdigit(s[i])){
            c3++;
        }
    }
    cout<<c1<<" "<<c2<<" "<<c3;
    return 0;
}

```

## 2. 合法的变量名？

### 题目描述

James在一节C++课程上，准备为自己的程序定义变量名称。老师告诉James，一个合法的变量名应该满足如下的三个条件：

- 1、只能由字母（大写或者小写）、数字及下划线（\_）组成
- 2、不能以数字开头
- 3、不能是C++中有特殊含义的单词，由于James是C++的初学者，只学过int、double、cout、cin这四个有特殊含义的单词，因此James只要避开这几个单词就可以。

请你编程帮助James判断他定义的变量名是否合法。

下表中列举了一些合法的变量名和非法的变量名的案例供你参考。

合法变量名案例	非法变量名案例
x	sum x: 有空格（只能是字母、数字或下划线）
sum	1x: 以数字开头
sum_x	int: 有特殊含义
sum2	a#: 有特殊字符“#”（只能是字母、数字、下划线）。
_sum	
INT注意：这是合法的，因为在C++中是区分大小写的，int是表示整数类型，但INT不是；	

### 输入

一行，包含一个字符串，是James为变量起的名字，且长度不大于20。

### 输出

一行，如果是合法的C++变量名，则输出yes，否则输出no。

### 样例输入

```
sum
```

## 样例输出

```
yes
```

## 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    bool f = true ;//假设合法
    getline(cin,s);
    //判断非法情况
    //1. 变量名除了字母数字下划线还有别的字符
    for(int i=0;i<s.size();i++){
        if((isalpha(s[i])||isdigit(s[i])||s[i]=='_')==false){
            f=false;
            break;
        }
    }
    //2. 判断是否开头为数字
    if(isdigit(s[0])){
        f=false;
    }
    //3. 含有 4个关键词
    if(s=="int"||s=="double"||s=="cin"||s=="cout"){
        f=false;
    }
    if(f){
        cout<<"yes";
    }else{
        cout<<"no";
    }
    return 0;
}
```

### 3. 重新排列

#### 题目描述

现在有一个20位以内的自然数，你可以将组成这个数的各位数字重新排列，得到一个数值为最小的新数，但新数的位数保持不变。请编程打印出重新排列后的新数。（如：231重新排序后位数不变的最小数是123，而23105重新排序后位数不变的最小数是10235。）

#### 输入

一个整数（位数  $\leq 20$ ）

#### 输出

重新排列后最小的新数

#### 样例输入

```
382
```

## 样例输出

## 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    /*
    对20位整数的每一位从小到大排序,
    如果第一个元素为'0',
    则将其与第一个非'0'元素交换
    */
    string s;
    cin>>s;
    sort(s.begin(),s.end());
    char t;//中转变量
    //如果首元素为'0', 就找到第一个非'0'的元素与其交换
    if(s[0]=='0'){
        for(int i=1;i<s.size();i++){
            if(s[i]!='0'){
                /*
                t=s[i];
                s[i]=s[0];
                s[0]=s[i];
                */
                swap(s[0],s[i]);
                break;
            }
        }
    }
    cout<<s;
    return 0;
}
```

**注意:** swap(a,b): 交换变量a和b的值

## 4.3 课后练习

### 1. 字符串分离

#### 题目描述

对输入的一行字符串做如下操作：将大写字母反向连成字符串，将小写字母正向连成字符串，最后输出一个字符串。

#### 输入

一行，包含若干个字符(个数不超过255，不含空格)

#### 输出

一行，包含所有分离后符合条件的字符连接成的字符串。

#### 样例输入

7DVesb#Ft%

## 样例输出

```
FVDesbt
```

### 2. 粉碎数字

#### 题目描述

小鱼同学得到了n个数字，他想知道，如果把这n个数字扔到粉碎机粉碎一下，能组成最大的数是多少？小鱼的同学们完全不明白粉碎机怎样粉碎数字，小鱼就数字粉碎做了一个定义：把数字完全打碎，比如有2个数198和63，那么粉碎后有数字1、9、8、6、3，那么能够组成最大的数就是98631。请你编写程序，帮助小鱼同学实现这个奇特的想法。

#### 输入

第一行是一个整数n (n是1~1000之间的整数)

第二行有n个整数 (每个整数都是0~9999之间的整数)

#### 输出

n个整数粉碎后能够组成的最大的整数

#### 样例输入

```
8  
1 89 654 750 4687 23 90 100
```

## 样例输出

```
99887766554432110000
```

### 3. n个一位数能够组成最大的数

#### 题目描述

请问n个一位数能够组成的最大的整数是多少。

比如，n=3，3个整数为1、3、9，那么组成最大的整数是931。

比如，n=4，4个整数为2、8、0、6，那么组成最大的整数是8620。

#### 输入

第一行为一个整数n (n<10)

第二行为n个一位数

#### 输出

一行，包含一个组成最大的整数。

#### 样例输入

```
4  
7 3 4 4
```

## 样例输出

```
7443
```

### 4. n个一位数能够组成最小的数

#### 题目描述

请问n个一位数能够组成的最小的n位整数是多少。

比如，n=3，3个整数为3、1、9，那么组成的最小整数是139。

比如，n=4，4个整数为2、8、0、6，那么组成的最小整数是2068。

### 输入

第一行为一个整数n (n<10)

第二行为n个一位数

### 输出

一行，包含一个组成的最小整数。

#### 样例输入

```
4
7 3 4 4
```

#### 样例输出

```
3447
```

## 5. string 和数值转换

### 5.1.1 相关函数

#### 1. 类型转换函数

1. stoi(s): 将字符串s转换为对应的整数

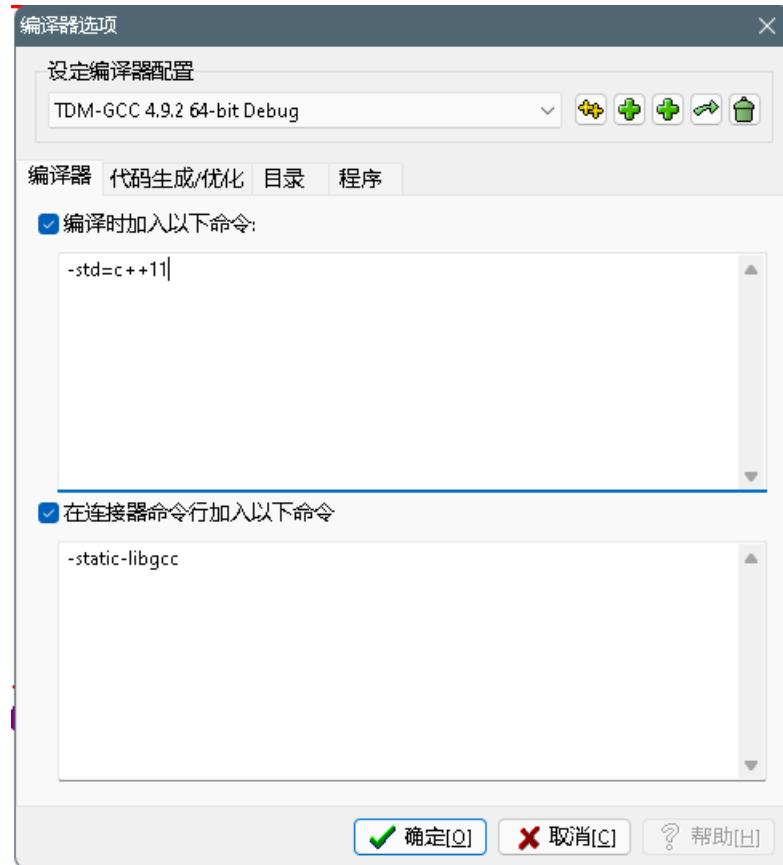
2. stoll(s): 将字符串s转换为对应的long long

3. stof(s): 将字符串s转换为对应的float

2. to\_string(int n): 将整数n转换为字符串

3. to\_string(double d): 将double型的d转换为字符串，转换成字符串小数点后6位

注：本功能需要C++11语法支持



**特别注意：**考NOIP的同学不要使用stoi等C++11的函数，需要改用atoi，用法如下：

```
string s = "123";
//s.c_str();将字符串s转为字符数组
int x = atoi(s.c_str());
cout<<x+10;
```

### 5.1.2 用法示例

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s = "100";
    int x = stoi(s);
    cout<<x+1<<endl;

    string s2 = "1234657899876540";
    long long l = stoll(s2);
    cout<<l<<endl;

    string s3 = "12.45467";
    float f = stof(s3);
    cout<<f<<endl;

    int a = 10,b = 20;
    string sa = to_string(a);
    string sb = to_string(b);
    cout<<sa+sb<<endl;
    return 0;
}
```

## 5.2 课堂案例

### 1. 整数串拆段

#### 题目描述

将一个长度小于10位的数字串拆成2段，使其和为最小的素数。

例如数字串‘13304’

拆的方法有：

1 + 3304 = 3305

13 + 304 = 317

133 + 04 = 137

1330 + 4 = 1334

从上面可看出，和为素数的有：317 与137，最小的是137

#### 输入

一个长度小于10的数字串

#### 输出

最小的和为素数的数，若无素数则输出 -1

#### 样例输入

```
13304
```

#### 样例输出

```
137
```

#### 思路：

将整数能够拆分的求和方案全部过一遍，以13304为例，5位数共有s.size()-1种也就是4种方案：

1 + 3304 = 3305

13 + 304 = 317

133 + 04 = 137

1330 + 4 = 13304

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s , s1 , s2;
    int x , y ;
    cin>>s;
    for(int i=0;i<s.size()-1;i++){
        s1 = s.substr(0,i+1);
        s2 = s.substr(i+1);
        // cout<<s1<<" "<<s2<<endl;
        x = stoi(s1);
        y = stoi(s2);
        cout<<x+y<<endl;
    }
    return 0;
}
```

第二步：判断结果是否为素数并找出最小值：

```
#include<bits/stdc++.h>
using namespace std;
bool sushu(int n){
    int i = 2;
    while(i<=sqrt(n)){
        if(n%i==0){
            return false;
        }
        i++;
    }
    return true;
}
int main(){
    int min = INT_MAX;
    string s , s1 , s2;
    int x , y ;
    cin>>s;
    for(int i=0;i<s.size()-1;i++){
        s1 = s.substr(0,i+1);
        s2 = s.substr(i+1);
        cout<<s1<<" "<<s2<<endl;
        x = stoi(s1);
        y = stoi(s2);
        cout<<x+y<<endl;
        if(sushu(x+y)&&(x+y)<min){
            min = x+y;
        }
    }
    if(min==INT_MAX){
        cout<<-1;
    }else{
        cout<<min;
    }
    return 0;
}
```

## 2. 趣味填空

### 题目描述

小华的寒假作业上，有这样一个趣味填空题：给出用等号连接的两个整数，如“1234 = 127”。当然，现在这个等号是不成立的。题目让你在左边的整数中间某个位置插入一个加号，看有没有可能让等号成立。以上面的式子为例，如果写成 $123+4=127$ ，这就可以了。请你编写一个程序来解决它。

### 输入

只有那个不相等的式子。已知，等号两边的整数都不会超过2000000000。

### 输出

如果存在这样的方案，请输出那个正确的式子。如果不存在解决方案，请输出“Impossible!”（引号中的部分）。

### 样例输入

```
1234=127
```

### 样例输出

```
123+4=127
```

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
/*
1324=127
第一步：将=两边的数据拆出来
left = 1234
right = 127
第二步：将left所有组合都列出来
第三步：判断等式是否成立
*/
string s ,l , r , s1 , s2 ;
int p , x , y , z ,i ;
bool f = false;
int main(){
    cin>>s;
    p=s.find("=");
    l=s.substr(0,p);
    r=s.substr(p+1);
    z=stoi(r);
    //cout<<l<<" "<<r;
    for(i=0;i<l.size()-1;i++){
        s1=l.substr(0,i+1);
        s2=l.substr(i+1);
        //cout<<s1<<" "<<s2;
        x=stoi(s1);
        y=stoi(s2);
        if(x+y==z){
            cout<<x<<"+"<<y<<"="<<z<<endl;
            f=true;
        }
    }
    if(!f){
        cout<<"Impossible!";
    }
    return 0;
}
```

## 5.3 课后作业

### 1. 简单a+b

#### 题目描述

张晓菲同学做了简单的ab求和的问题。但是，如果要求输入的情况不是a和b，而是整个加法表达式呢？请想办法，计算加法表达式的结果。

#### 输入

输入一个加法表达式，如 $1+2=$ ，或者 $23+58=$ 。（注意：做加法的2个整数都在0~109的范围内）

### 输出

计算出输入表达式的正确结果

### 样例输入

```
1+2=
```

### 样例输出

```
3
```

## 2. 简单 $a*b$

### 题目描述

按照 $a*b=$ 的格式输入算式，通过计算输出 $a*b$ 的结果。

### 输入

输入中包括一个表达式，如： $a*b=$  a和b都是int类型的正整数。

### 输出

结果只有一个正整数，整数在long long范围内。

### 样例输入

```
100*200=
```

### 样例输出

```
20000
```

## 3. 表达式的值(III)

### 题目描述

从键盘读入一个格式为 $(a+b)*c$ 的运算式，请计算该运算式的结果。

如：输入 $(12+13)*20$ ，输出：500

(4.2.62)

### 输入

一个如 $(a+b)*c$ 的运算式（本题测试数据中a、b、c都是int范围的数，计算结果也在int范围）

### 输出

一个整数代表运算结果

### 样例输入

```
(12+13)*20
```

### 样例输出

```
500
```

## 5.4 string 常见函数对照表

函数	举例说明
size()	求字符串长度，等于length()函数
s[下标]	获取字符串的某个字符，等同于at()函数： s="abcd"; cout<<s[0]<<s.at(2)<<endl;
getline(cin,s)	读入一整行（直到换行），包含输入的空格
find(子串 substr)	查找子字符串第一次出现的下标，没有返回string::npos(注意判断其为-1)，如： s="abcd"; cout<<s.find("cd")<<endl; 结果：2 如： string s = "abcd"; cout<<(s.find("hello") == -1)<<endl; 结果：1（结果为true则输出1）
find(substr,x)	在字符串的第x个位置后，查找子串substr
substr(开始位置i, 子串长度 len)	截取子字符串，但len>字符串长度的时候，只取剩余的。 注意：i要在下标范围内： string s="abcdefg"; cout<<s.substr(3,2)<<s.substr(3,20); 结果：dedefg 注意：如果长度大于字符串长度，只取到最后
erase(开始位置i,删除长度len)	删除字符串中第i个位置开始的len个字符 s="abcdef"; s.erase(2,3); cout<<s<<endl; 结果是：abf 注意：如果len大于字符串长度，则删到最后

erase(开始位置i,删除长度len)	删除字符串中第i个位置开始的len个字符 s="abcdef"; s.erase(2,3); cout<<s<<endl; 结果是：abf 注意：如果len大于字符串长度，则删到最后
insert(插入位置i, 插入字符串 s)	在字符串第i个位置插入一个字符串s s="abcdef"; s.insert(2,"+"); cout<<s<<endl; 结果：ab+cdef
replace(开始位置i,长度len,要替换的字符串 ss)	用字符串ss替换字符串中i开始长度是len的一段，如： s="abcdef"; s.replace(2,1,"123"); cout<<s; 结果是：ab123edf
字符判断	isalpha():判断是否为字母 islower():判断是否为小写 isupper():判断是否为大写 isdigit():判断是否为数字
字符转换	tolower(s[i]):转小写 toupper(s[i]):转大写
获取头尾指针	范围s的头尾指针，和数组指针一致。 s.begin();s.end();用于排序函数sort()。 sort(s.begin(),s.end()); reverse(s.begin(),s.end());
字符串转整数、长整数、浮点数	int : stoi("123") long long : stoll("123456789987"); float : stof("12.3456")

	<p>to_string(int n)功能：将数字转换成字符串 to_string(double d)功能：将实数转换成字符串，转换成的字符串小数点后6位 本功能需要c++11的语法支持</p> 
特别注意	<p>考NOIP的同学不要使用stoi等C++11的函数，需要改用atoi，用法如下：</p> <pre>string s = "123"; //s.c_str();将字符串s转为字符数组 int x = atoi(s.c_str()); cout&lt;&lt;x+10;</pre>

## 第二十三章 string 的进阶问题

### 1.课堂练习

#### 1. 词组缩写

##### 题目描述

定义：一个词组中每个单词的首字母的大写组合称为该词组的缩写。 比如， C语言里常用的EOF就是end of file的缩写。

##### 输入

测试数据占一行，有一个词组，每个词组由一个或多个单词组成；每组的单词个数不超过10个，每个单词有一个或多个大写或小写字母组成； 单词长度不超过10，由一个或多个空格分隔这些单词。

##### 输出

输出规定的缩写

##### 样例输入

```
end of file
```

##### 样例输出

```
EOF
```

##### 思路：

每个单词都由一个或多个大写或小写字母组成并且由一个或多个空格隔开。

第一步：

找出每个单词的首字母：

1. 如果第一个是字符且不是空格或者（当前不是第一个字符且不是空格并且上一个字符是空格）

2. 从第二个字符开始循环，如果当前字符不是空格且上一个字符是空格则是首字母

第二步：

如果是小写就转成大写

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    char t[1];
    string s, x;
    getline(cin,s);
    int i;
    for(i=0;i<s.size();i++){
        if (i==0&&s[i]!=' '|| i>0 && s[i]!=' ' && s[i-1]==' ') {
            t[0]=toupper(s[i]);
            x+=t[0];
        }
    }
    cout<<x;
    return 0;
}
```

## 2. 字符串压缩

### 题目描述

输入字符串，输出压缩后的字符串。压缩的方法是把连续的相同字母压缩为"长度+字母"的形式，在本题中，单个的字母不需要压缩。

### 输入

一行，一个字符串,只包含小写英文字母,长度不超过255。

### 输出

### 样例输入

```
aaabbbbbx
```

### 样例输出

```
3a5bx
```

### 思路

采用数数的思想，数出连续相同的字符有几个。

c=0;

每遇到一个字符就c++，并判断连续字符是否结束，如果结束，输出字符及出现的次数并清空计数器；

即：

if(到了最后一个字符或者当前字符!=下一个字符)

## 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    int i,c=0;
    cin>>s;
    for(i=0;i<s.size();i++){
        c++;
        if(i==s.size()-1 || s[i]!=s[i+1]){
            if(c!=1){
                cout<<c<<s[i];
            }else{
                cout<<s[i];
            }
            c=0;
        }
    }
    return 0;
}
```

## 补充：

- 1) 如果 || 第一个条件为true，则第二个条件不判断，直接认定结果为true
  - 2) 如果 && 第一个条件为false，则第二个条件不判断，直接认定结果为false
3. 我是第几个单词

### 题目描述

输入一个英文句子，例如：“This is a Book.”，可以看到句子是以“.”来作为结束符号的，并且单词之间以一个空格来分隔。接着再输入一个单词A，请找出首次在句子中出现的与A相同的单词，是句子中的第几个单词，若不存在，则输出该句子中单词字符的总个数。例如对上句子而言，若输入单词“is”，则应输出：2 若输入单词“isa”，则应输出：11

### 输入

第一行为以‘.’结束的一个词组（仅由若干个单词组成，单词间由一空格隔开，除单词和最后的“.”以外，不含其它字符）

第二行是一个单词（不含空格）

### 输出

一个整数

### 样例输入

```
This is a Book.  
Book
```

### 样例输出

4

## 思路：

第一步：分解字符串中的每个单词，并输出每个单词是第几个。判断依据：当前如果不是空格，则一定是单词的一部分，存入w。例如：

this is a book.

this	1
is	2
a	3
book	4

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
string s,w,t;
int main(){
/*
以“.”作为结束符号
并且单词之间以一个空格来分隔
找出句子中首次出现的单词
*/
int c=0,i;
getline(cin,s);
cin>>t;//要查找的单词
for(i=0;i<s.size();i++){
    if(s[i]!=' '){
        w+=s[i];
        if(s[i+1]==' '||s[i+1]=='.'){
            c++;
            cout<<w<<" "<<c<<endl;
            w="";
        }
    }
}
return 0;
}
```

第二步：判断每个单词是否是我们要找的那个，如果是，则输出并break，不是则输出总字符数。

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
string s,w,t;
int main(){
/*
以“.”作为结束符号
并且单词之间以一个空格来分隔
找出句子中首次出现的单词
*/
int c=0,i,sum=0;
bool f=false;
getline(cin,s);
cin>>t;//要查找的单词
for(i=0;i<s.size();i++){
    if(s[i]==t[0] && f==false){
        sum+=c;
        f=true;
    }
}
cout<<sum;
}
```

```

    if(s[i]!=' '){
        w+=s[i];
        if(s[i+1]==' '||s[i+1]=='.'){
            c++;
            sum+=w.size();
            //cout<<w<<" "<<c<<endl;
            if(w==t){
                cout<<c;
                f=true;
                break;
            }
            w="";
        }
    }
}
if(!f){
    cout<<sum;
}
return 0;
}

```

#### 4. 表达式的值

##### 题目描述

给出一个表达式,其中运算符仅包含+,要求求出表达式的最终值。

如: 1+1, 则结果为2, 1+2+3则结果为6, 12+23则结果35。

##### 输入

仅一行, 即为含有正整数和+号的表达式。

##### 输出

仅一行, 既为表达式算出的结果 (所有数据保证计算结果<=100000000)。

##### 样例输入

1+1

##### 样例输出

2

##### 思路

第一步, 分解并输出每个连续的数字

```

#include<bits/stdc++.h>
using namespace std;
string s,w;
int main(){
    int x,r,i;
    getline(cin,s);
    for(i=0;i<s.size();i++){
        //如果当前字符是数字, 则肯定是连续数字的一部分
        if(isdigit(s[i])){
            w+=s[i];
            //判断是否结束
        }
    }
}

```

```

        if(i==s.size()-1 || !isdigit(s[i+1])){
            cout<<w<<endl;
            w="";
        }
    }
}

return 0;
}

```

第二步，将数字转换为整数，叠加起来：

```

#include<bits/stdc++.h>
using namespace std;
string s,w;
int main(){
    int x,r,i,sum=0;
    getline(cin,s);
    for(i=0;i<s.size();i++){
        //如果当前字符是数字，则肯定是连续数字的一部分
        if(isdigit(s[i])){
            w+=s[i];
            //判断是否结束
            if(i==s.size()-1 || !isdigit(s[i+1])){
                //cout<<w<<endl;
                sum+=stoi(w);
                w="";
            }
        }
    }
    cout<<sum;
    return 0;
}

```

## 5. 分数计算

### 题目描述

从键盘读入一个分数算式，为2个分数做加法或者减法，请输出分数算式的结果，结果也用分数表达，且约分到最简形式。（请注意：做减法可能得到负的分数，如果是负数要输出负号-，如 $1/15 - 4/15$ 结果为 $-1/5$ ）

### 输入

分数表达式（分数表达式中，每个分数的分子和分母都是正整数，两个分数中的运算符，可能是加号，也可能是减号，且分数表达式不含空格）

### 输出

分数表达式计算的结果

### 样例输入

1/12+5/12

### 样例输出

1/2

## 提示

注意考虑特殊情况,如: $1/2+1/2=1$ , $1/2-1/2=0$ ,这些情况下结果不需要表现为分数形式。

## 思路

第一步：定义函数求出两个整数的最大公约数

**辗转相除法，求最大公约数：**

反复使用 $a \% b$ , 如果 $a \% b == 0$ , 则 $b$ 就是最大公约数, 如果 $a \% b != 0$ ,  $a$ 取 $b$ 的值,  $b$ 取余数的值:

$a \% b = 8 \% 12 = 8$

$12 \% 8 = 4$

$8 \% 4 = 0$

求 $a$ 和 $b$ 最小公倍数, 先求出最大公约数为 $t$ , 最小公倍数= $a * b / t$ ;

第二步：模拟分数计算的过程直接计算

分解分子、分母, 进行计算, 然后月份, 判断特殊情况 (分子是分母的倍数)

**解法一：**将分数表达式作为字符串读入, 然后模拟计算

```
#include<bits/stdc++.h>
using namespace std;
int gys(int a,int b){//求公约数
    int t;
    while(a%b!=0){
        t=a%b;
        a=b;
        b=t;
    }
    return b;
}

//s:总表达式 s1:左边分数 s2: 有变分数
//fz1,fm1: 第一个分数的分子分母
string s,s1,s2,fz1,fm1,fz2,fm2;
//整数存放分子和分母
int a1,b1,a2,b2;
int p;//存放运算符, +或-的下标
int r1,r2;//存放计算结果的分子和分母
int t;
int main(){
    cin>>s;
    //1/12+5/12
    p=s.find("+");
    if(p==1){ //说明是减法
        p=s.find("-");
    }
    s1=s.substr(0,p);
    s2=s.substr(p+1);
    // cout<<s1<<" "<<s2;
    fz1=s1.substr(0,s1.find("/"));
    fm1=s1.substr(s1.find("/") + 1);

    fz2=s2.substr(0,s2.find("/"));
    fm2=s2.substr(s2.find("/") + 1);
```

```

// cout<< fz1<<" "<< fm1<<" "<< fz2<<" "<< fm2<<"\n";
a1=stoi(fz1);
b1=stoi(fm1);
a2=stoi(fz2);
b2=stoi(fm2);
//计算结果的分母
r2=b1*b2;
//计算结果的分子
if(s.find("+") != -1){
    r1=a1*b2+a2*b1;
}else{
    r1=a1*b2-a2*b1;
    if(r1<0){
        cout<<"-";
        r1=r1*-1;
    }
}
// cout<<r1<<"/"<<r2;
t=gys(r1,r2);
//如果分子是分母的倍数，直接除掉
if(r1%r2==0){
    cout<<r1/r2;
}else{
    cout<<r1/t<<"/"<<r2/t;
}
return 0;
}

```

**解法二：**根据分数表达式的特性，直接读入分子和分母，简化截取分子、分母及转换分子分母的过程

```

#include<bits/stdc++.h>
using namespace std;
int gys(int a,int b){ //求公约数
    int t;
    while(a%b!=0){
        t=a%b;
        a=b;
        b=t;
    }
    return b;
}

int a1,b1,a2,b2;
int p; //存放运算符的下标
int r1,r2; //存放计算结果的分子、分母
int t;
char c,f;
int main(){
    cin>>a1>>c>>b1>>f>>a2>>c>>b2;
    //计算结果的分母
    r2 = b1*b2;
    //计算结果的分子
    if(f=='+' ){
        r1 = a1*b2+a2*b1;
    }
}

```

```
    } else{
        r1=a1*b2-a2*b1;
        if(r1<0){
            cout<<"-";
            r1=r1*-1;
        }
    }
    t=gys(r1,r2);
    if(r1%r2==0){
        cout<<r1/r2;
    }else{
        cout<<r1/t<<"/"<<r2/t;
    }
    return 0;
}
```

## 2.进阶问题课后作业

### 1. 统计单词个数

#### 题目描述

输入一行字符串，包含若干个单词，约定相邻的两个单词用空格隔开（一个或多个空格），编程统计单词的个数

#### 输入

一行空格隔开的若干个单词。

#### 输出

单词个数

#### 样例输入

```
Hello  world
```

#### 样例输出

```
2
```

### 2. 找最长单词

#### 题目描述

编写程序，根据给出的一个结束于'.'的字符字串，找出其中最长的含有字母'a'的子串。

#### 输入

一行，为一个字符字串，结束于句点'.'。字串中的子串由一个或几个空格隔开。

#### 输出

一行。显示找出的最长的含有字母'a'的子串。如果有多个这样的子串，只显示其中的第一个；若没有含字母'a'的子串，则显示'NO'。

#### 样例输入

```
Her name is Lilan and she is a happy student.
```

#### 样例输出

### 3. 求英文句子中的最长单词

#### 题目描述

一个英文句子（长度不超过255），只含有字母和空格，输出最长的一个单词。如有多个长度相同的单词，则输出最前面的一个。

#### 输入

一个字符串。

#### 输出

一个字符串。

#### 样例输入

```
in which four coins
```

#### 样例输出

```
which
```

### 4. 隐藏的最大整数

#### 题目描述

今天是个好日子，整数小伙伴们又一起出来聚会了。大家商议决定：今天玩捉迷藏！ 玩法很简单，就是把藏在一个长长的字符串中（任何两个人都不会并排藏在一起）数字伙伴找出来。而且，因为小伙伴们太多，只找到此次藏起来的最大的那个伙伴就可以了。并且，大家一致同意，让“1”做第一个找人的人——当然，事先，他并不知道到底有哪些伙伴藏起来了…… 游戏开始了。

“1”是一个很聪明的人，很快，他就把最大的那个伙伴找出来了…… 如果你是“1”，你能写个程序来解决这件事情么？

请注意：隐藏在字符串中的整数不会以0开头，也就是不存在这种字符串\*a032AB342，且读入的字符串不含空格。

#### 输入

只有一个字符串，这里面藏有很多的整数小伙伴。（字符串长度≤100，且不含空格）

#### 输出

只有一个整数，表示藏在其中的最大的那个整数小伙伴的位置（整数第一个数字在原串中的位置）。

#### 样例输入

```
*((*-a32AB342+//32143abAA
```

#### 样例输出

```
17
```

### 5. 字符串解压

#### 题目描述

输入压缩后的字符串，输出压缩前的字符串。压缩的方法是把连续的相同字母压缩为"长度+字母"的形式，在本题中，单个的字母不需要压缩。例如：3a5bx，解压后的结果为：aaabbbbbx；例如：12ab10c2ax解压后的结果为：aaaaaaaaaabcccccccaax。

### 输入

压缩后的字符串

### 输出

解压后的字符串

### 样例输入

```
3a5bx
```

### 样例输出

```
aaaaaaaaaabcccccccaax
```

## 6. 保留整数

### 题目描述

输入一个字符串str1，把其中的连续非数字的字符子串换成一个'\*'，存入字符数组str2 中，所有数字字符也必须依次存入 str2 中。输出str2。

### 输入

输入为一行字符串str1，其中可能包含空格。字符串长度不超过80个字符。

### 输出

输出处理好的字符串str2。

### 样例输入

```
$Ts!47&*s456 a23* +B9
```

### 样例输出

```
*47*456*23*9
```

## 7. 计算表达式

### 题目描述

表达式的形式如：3+56-4 其中，运算数为一位整数，运算符为 +、-、\* 三种，且运算符没有优先级的区分，一律自左向右计算。如上例的计算过程为： $3+56-4=86-4=48-4=44$

### 输入

一行，即表达式字符串（长度小于100）

### 输出

一个整数，即表达式的计算结果（结果在-20000至20000之间）

### 样例输入

```
3+5*6-4
```

## 样例输出

44

### 8. 表达式的值II

#### 题目描述

给出一个表达式，其中运算符仅包含+和-两种运算符，且没有括号。要求求出表达式的最终值。

数据保证要计算的数字的数量不超过100个，每个整数的数值和计算结果的数值都在int的范围内。

请注意：数字可能是1位数，也可能是多位数，如： $1+1-1=1$ ， $12+23-11=24$ 。

#### 输入

仅一行，即为表达式。

#### 输出

计算结果的数值。

#### 样例输入

1+1-1

## 样例输出

1

### 9. 求多个分数的和

#### 题目描述

从键盘读入一个求和算式，求出多个分数的和，结果也用分数表达，并约分到最简形式。

如，从键盘读入 $1/12+5/12+1/3$ ，则输出： $5/6$

(5.1.107)

#### 输入

分数求和表达式

#### 输出

和的分数形式

#### 样例输入

1/12+5/12+1/3

## 样例输出

5/6

#### 提示

注意考虑多个分数的和为整数的情况如： $1/3+1/3+1/3=1$

# 第二十四章 进制转换

## 1.N进制和10进制的互转

### 1.1 进制的基础知识

#### 1. 什么是进制?

进制也就是进位计数制。是人为定义的带进位的计数方法。对于任何一种进制X进制，就表示每一位置上的数运算都是逢X进1位。十进制是逢十进一，十六进制是逢十六进一，以此类推。

#### 2. 生活中的进制有哪些?

10进制，60进制，12进制，24进制等等。

#### 3. n进制如何数数?

10进制: 0 1 2 3 4 5 6 7 8 9 10 11...

2进制: 0 1 10 11 100 101 110...

8进制: 0 1 2 3 4 5 6 7 10 11 12 13 14 15 16 17 20 21...

16进制: 0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12  
13 14 15 16 17 18 19 A...

### 1.2 十进制转为R进制

#### 1. 十进制整数转换成 R 进制的整数: 除R 取余法。

2. 学习技巧: 可以参考短除法, 短除法的作用是用来拆出10进制n的每一位, 可以理解为10进制转为10进制, 使用方法是除10取余然后倒过来:

例:  $(10)_{10} = (1010)_2$

2 | 10  
2 | 5 0  
2 | 2 1  
2 | 1 0  
0 1

#### 3.课堂练习

$$(12)_{10} = (\underline{\hspace{2cm}})_2$$

$$(126)_{10} = (\underline{\hspace{2cm}})_8$$

$$(258)_{10} = (\underline{\hspace{2cm}})_{16}$$

## 1.3 R进制转为十进制

### 1. 按权展开！

基数为 R 的数字，只要将各位数字与它的权相乘，其积相加求出的和数就是十进制数。

例：

$$3506.2_8$$

$$=6 \times 8^0 + 0 \times 8^1 + 5 \times 8^2 + 3 \times 8^3 + 2 \times 8^{-1}$$

$$=1862.25$$

$$0.2A_{16}$$

$$=2 \times 16^{-1} + 10 \times 16^{-2}$$

$$=0.1640625$$

### 2. 学习技巧：可以参考10进制整数计算机制来学习

$$12345 = 5 * 1 + 4 * 10 + 3 * 100 + 2 * 1000 + 1 * 10000$$

$$= 5 * 10^0 + 4 * 10^1 + 3 * 10^2 + 2 * 10^3 + 1 * 10^4$$

### 3. 课堂练习

R进制转10进制实例

$$1100_2 = 0 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 = (12)_{10}$$

$$3506_8 = 6 \times 8^0 + 0 \times 8^1 + 5 \times 8^2 + 3 \times 8^3 = (1862)_{10}$$

$$1A_{16} = 10 * 16^0 + 1 * 16^1 = (26)_{10}$$

### 4. 作业

$$(110011)_2 = (\underline{\hspace{2cm}})_{10}$$

$$(267)_8 = (\underline{\hspace{2cm}})_{10}$$

$$(2AF)_{16} = (\underline{\hspace{2cm}})_{10}$$

## 1.4 十进制和R进制相互转换的程序实现

### 1. 正整数N转换成一个二进制数

#### 题目描述

输入一个不大于32767的整数n，将它转换成一个二进制数。

#### 输入

输入只有一行，包括一个整数n(0<=n<=32767)

#### 输出

输出只有一行。

#### 样例输入

## 样例输出

```
1100100
```

## 思路：

1. 定义字符串存储N转换的二进制数
2. 用短除法除2取余逆序存储字符串

## 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s ;
    int n,x;
    char c;
    cin>>n;
    while(n!=0){
        x = n%2;
        c = x + '0';
        s = c + s ;
        n/=2;
    }
    if(s==""){
        cout<<0;
    }else{
        cout<<s;
    }
    return 0;
}
```

## 2. 二进制转换十进制

### 题目描述

请将一个25位以内的2进制正整数转换为10进制！

### 输入

一个25位以内的二进制正整数

### 输出

该数对应的十进制

## 样例输入

```
111111111111111111111111111111
```

## 样例输出

```
16777215
```

## 思路：

从最低位开始 (s.size()-1),倒过来计算 (按权展开)

## 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    int r,t=1,i;//t表示权重
    cin>>s;
    for(i=s.size()-1;i>=0;i--){
        r=r+(s[i]-'0')*t;
        t=t*2;
    }
    cout<<t;
    return 0;
}
```

### 3. 正整数n转换为16进制

#### 题目描述

请从键盘读入一个非负整数n (n是一个不超过18位的正整数) , 将n转换为16进制!

注意: 16进制即逢16进1, 每一位上可以是从小到大为0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F共16个大小不同的数, 即逢16进1, 其中用A, B, C, D, E, F这六个字母来分别表示10, 11, 12, 13, 14, 15。

如: 60的十六进制为3C。 (字母请用大写)

#### 输入

一个不超过18位的非负整数n

#### 输出

该数的十六进制值

#### 样例输入

```
1000000000000
```

#### 样例输出

```
174876E800
```

#### 思路:

除16取余。

逆序存储到字符串时需要注意:

整数 $0_9$ 转换字符'0' '9'是+'0'

整数 $10_{15}$ 转换字符'A' 'F'是+'A'-10

#### 示例代码

## 1.5 可选练习

## 2.二、八、十六进制互转

1. 2、8、16进制的互相转换的方法
2. 6.2 2、8、16进制的互相转换的实现

## 3.课后作业

## 第二十五章 高精度计算

### 1.高精度计算的基础知识

#### 1.1 什么是高精度运算？

1. C++的基本类型和范围

类型	定义	大小	范围
char	字符型	1 byte	-128~127
short(int)	短整型	2 byte	-32768~32767 ( $-2^{15}$ ~ $2^{15}-1$ )
int	整形	4 byte	-2147483648 ~ 2147483647 ( $-2^{31}$ ~ $2^{31}-1$ )
long	长整形	4 byte(32bit 系统) 或 8 byte(64bit 系统)	参考 int 和 long long 的范围
long long	长整形	8 byte	-9223372036854775808 ~ 9223372036854775807 ( $-2^{63}$ ~ $2^{63}-1$ )
float	实型	4 byte	- (10 的 38 次方) ~ 10 的 38 次方
double	双精度	8 byte	- (10 的 308 次方) ~ 10 的 308 次方

1 byte = 8 位

2. 什么是高精度计算？

高精度运算是指参与运算的数（加数，减数，因数等）范围大大超过了标注数据类型能表示的范围的计算。eg：求两个300位数的积，常规方法显然无法满足需求，这时就需要高精度计算。

#### 1.2 高精度计算的基本思路

##### 基本思路：

1. 由于字符数组可以输入任意位，因此采用字符串（或者字符数组）读入两个高精度的数；
2. 由于加减乘运算都需要从右向左运算（包括进位），而且需要进行整数运算，因此为了方便，我们将2个字符数组逆序存入2个整型数组，这样既可以从左向右计算，又可以按照整数格式进行运算，比较方便；

3. 将计算结果存入第三个数组，然后按照要求逆序输出结果，就可以实现高精度运算。

**注意：**考虑高精度减法、乘法中运算结果为0的情况。

## 2. 高精度计算的实现

### 2.1 课堂案例

#### 1. 【基础】高精度加法

##### 题目描述

计算 $a+b$ 的值， $a,b$ 皆为不超过240位的正整数。

##### 输入

两个正整数，每行一个

##### 输出

一个数，代表两个整数的和

##### 样例输入

```
111111111111111111111111111111111111111111111  
222222222222222222222222222222222222222222222
```

##### 样例输出

```
333333333333333333333333333333333333333333333
```

##### 思路：

以 $968+94$ 为例，数学运算的写法是：

$$\begin{array}{r} & 9 & 6 & 8 \\ & + & 9 & 4 \\ \hline \text{做加法} & 9 & 15 & 12 \\ \text{进位得结果} & 1 & 0 & 6 & 2 \end{array}$$

在高精度计算中，为了方便模拟右对齐、从右向左做加法，以及从右向左进位的过程：

1. 将2个数逆序、逐位存入2个整数数组，这样就能实现左对齐；
2. 从左向右做加法及进位

```
s1 = "968"  
s[0]——>a[s1.size()-1]  
s[1]——>a[s1.size()-2]  
s[i]——>a[s1.size()-i-1]
```

##### 示例代码：

```
#include<bits/stdc++.h>  
using namespace std;  
int main(){  
    /*  
     * 第一步：用string读入  
     * 第二步：将两个高精度整数逆序存入ab两个整数数组  
     * 第三步：从左向右，逐位求和，结果存入c数组，从左向右逐位进位  
    */
```

```

第四步：逆序输出结果
*/
string s1,s2;//高精度数
int a[250]={0},b[250]={0},c[500]={0};
int i,len;
cin>>s1>>s2;
//将两个高精度整数逆序存入ab两个整数数组
for(i=0;i<s1.size();i++){
    a[s1.size()-1-i]=s1[i]-'0';
}
for(i=0;i<s2.size();i++){
    b[s2.size()-1-i]=s2[i]-'0';
}
//从左向右，逐位求和,
//结果存入c数组，从左向右逐位进位
//运算的速度取决于两个整数中较长的
if(s1.size()>s2.size()){
    len=s1.size();
} else{
    len=s2.size();
}
//逐位相加
for(i=0;i<len;i++){
    c[i] = a[i]+b[i];
}
//逐位进位
for(i=0;i<len;i++){
    if(c[i]>=10){
        c[i+1]=c[i+1]+c[i]/10;//本题中加1效果一样
        c[i]=c[i]%10;
    }
}
//逆序输出结果
//两个不超过len位的整数做加法，结果可能是len+1
if(c[len]!=0){
    len++;
}
for(i=len-1;i>=0;i--){
    cout<<c[i];
}
return 0;
}

```

## 2. 【基础】高精度减法

### 题目描述

高精度减法，求 $a-b$ 。 $a,b$ 都是不超过240位的非负整数。

### 输入

两个非负整数，每行一个。

### 输出

一个整数，代表两个整数相减之后的结果。

### 样例输入

333  
222

样例输出

**思路：**

如果 $a>b$ , 结果为正, 用 $a-b$ , 如果 $a< b$ , 结果为负, 交换 $ab$ 的值, 继续使用 $a-b$ 。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
/*
第一步：判断正负
第二步：将两个字符逆序存入整数数组
第三步：从左向右，逐位相减，不够借位
第四步：逆序输出结果
*/
string s1,s2;//高精度数
int a[250]={0},b[250]={0},c[500]={0};
int i,len,p;
char f = '+';
cin>>s1>>s2;
//字符串长的一定大，一样长的话字典码大的一定大
if(s1.size()<s2.size() || (s1.size()==s2.size() && s1<s2)){
    f='-';
    swap(s1,s2);//交换值
}
//存入数组
for(i=0;i<s1.size();i++){
    a[i]=s1[s1.size()-1-i]-'0';
}
for(i=0;i<s2.size();i++){
    b[i]=s2[s2.size()-1-i]-'0';
}
len = s1.size();
for(i=0;i<len;i++){
    if(a[i]<b[i]){
        a[i+1]=a[i+1]-1;
        a[i]=a[i]+10;
    }
    c[i]=a[i]-b[i];
}
if(f=='-'){
    cout<<f;
}
//找到第一个不是0的最高位
for(i=len-1;i>=0;i--){
    if(c[i]!=0){
        p=i;
        break;
    }
}
```

```
    }
    for(i=p;i>=0;i--){
        cout<<c[i];
    }
    return 0;
}
```

### 3. 【基础】高精度乘单精度

## 题目描述

高精度乘单精度， $a \times b$ 。  $a$ 是一个很大的非负整数，但不超过240位， $b$ 是一个非负整数不超过10000，求 $a \times b$ 。

输入

两行数字，第一行是a，第二行是b。

输出

一行，输出 $a * b$ 的计算结果。

## 样例输入

## 样例输出

思路：

123×45可以用5和40分别乘以123，然后相加；也可以用45分别乘以100,20,3，然后相加。

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s1;
    int a[250]={0},c[500]={0};
    int i,j,p=0,b;//p=0防止出现乘数为0的情况
    cin>>s1>>b;
    //存入数组
    for(i=0;i<s1.size();i++){
        a[i]=s1[s1.size()-i-1]-'0';
    }
    //逐位相乘
    for(i=0;i<s1.size();i++){
        c[i]=a[i]*b;
    }
    //进位
    for(i=0;i<s1.size();i++){
        if(c[i]>=10){
            c[i+1]=c[i+1]+c[i]/10;
            c[i]%=10;
        }
    }
}
```

```

//找出第一个非0元素并逆序输出
for(i=s1.size()-4-1;i>=0;i--) {
    if(c[i]!=0) {
        p=i;
        break;
    }
}
for(i=p;i>=0;i--) {
    cout<<c[i];
}
return 0;
}

```

#### 4. 【基础】高精度乘

##### 题目描述

高精度乘，求两个很大的非负整数相乘的结果。

##### 输入

2个非负整数，每个一行，每个整数不超过240位。

##### 输出

一个整数，表示相乘的结果。

##### 样例输入

```

1111111111111111111111
2222222222222222222222

```

##### 样例输出

```
2469135802469135802469135308641975308641975308642
```

思路：

### 乘的数学实现

$$\begin{array}{r}
 & 1 & 2 & 5 \\
 & * & & \\
 & 6 & 2 & 5 \\
 \hline
 \text{逐位相乘 (进位)} & 2 & 5 & 0 \\
 \text{错位相加+} & 3 & 1 & 2 & 5
 \end{array}$$

## 高精度模拟实现

下标	0	1	2	3	
*	5	2	1		数组a
逐位乘 (进位)	5	2			数组b
逐位乘 (进位)	5	2	6		数组c
错位相加+	5	2	1	3	
逆序输出结果	3	1	2	5	数组c

j=0 j=1 j=2

i=0: 5 2 6

c[0] c[1] c[2]

i=1: 0 5 5

c[1] c[2] c[3]

c[j+i]

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s1,s2;
    int a[250]={0},b[250]={0},c[500]={0};
    int i,j,p=0;
    cin>>s1>>s2;
    //存入数组
    for(i=0;i<s1.size();i++){
        a[i]=s1[s1.size()-i-1]-'0';
    }
    for(i=0;i<s2.size();i++){
        b[i]=s2[s2.size()-i-1]-'0';
    }
    //循环a数组的每一位，用a[i]乘以b数组上的每一位b[i]
    //结果错位相加到c数组的c[i+j]位上
    for(i=0;i<s1.size();i++){
        for(j=0;j<s2.size();j++){
            c[i+j]=c[i+j]+a[i]*b[j];
            if(c[i+j]>=10){
                c[i+j+1]=c[i+j+1]+c[i+j]/10;
                c[i+j]=c[i+j]%10;
            }
        }
    }
    //逆序输出
    for(i=s1.size()+s2.size()-1;i>=0;i--){
        if(c[i]!=0){
```

```
    p=i;
    break;
}
}
for(i=p;i>=0;i--){
    cout<<c[i];
}
return 0;
}
```

## 5. 【基础】高精度整数除法

### 题目描述

求a/b的结果。已知a, b为 $10^8$ 范围内的非负整数，求a/b保留前n位小数商的结果。

(5.1.72)

### 输入

a b n

### 输出

一行数字

### 样例输入

```
97 61 50
```

### 样例输出

```
1.59016393442622950819672131147540983606557377049180
```

### 思路：

除法整数位就是a/b，第一个小数位是 $a \% b * 10 / b$ ，以此类推。

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a,b,n,i,t;
    cin>>a>>b>>n;
    cout<<a/b<<"." ;
    t=a%b;
    for(i=0;i<n;i++){
        cout<<t*10/b;
        t=t*10%b;
    }
    return 0;
}
```

## 6. 【基础】求2的n次方

### 题目描述

求2的n次方！(0<=n<=100)

### 输入

从键盘读入一个整数n!

### 输出

请输出2的n次方！

### 样例输入

```
100
```

### 样例输出

```
1267650600228229401496703205376
```

### 思路：

准备一个整数数组a，存放2的n次方，a数组默认存储一个1，代表2的0次方。

循环n次，每次循环都要将a数组的每一位乘以2，并进位。然后判断a数组乘以2是否多出一位，如果多一位，那么a数组的位数计数器要+1.

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int a[100]={1};
int i,j,k=1,n;
int main(){
    cin>>n;
    //循环n次，每次都将a数组*2
    for(i=1;i<=n;i++){
        //将数组每一位都*2
        for(j=0;j<k;j++){
            a[j]=a[j]*2;
        }
        //逐位进位
        for(j=0;j<k;j++){
            if(a[j]>=10){
                a[j+1]=a[j+1]+a[j]/10;
                a[j]=a[j]%10;
            }
        }
        //判断a数组是否多出一位
        if(a[k]!=0){
            k++;
        }
    }
    //逆序输出
    for(i=k-1;i>=0;i--){
        cout<<a[i];
    }
    return 0;
}
```

## 7. 【基础】求 $2+2^2+2^2\cdot 2+\dots+2^2\cdot 2^2\cdots \cdot 2^2$

### 题目描述

求 $2+2^2+2^2\cdot 2^2+\dots+2^2\cdot 2^{\cdot 2^{\cdot \dots^{\cdot 2}}}$ 的和是多少？最后一项有多少2相乘由键盘读入的n决定  
( $1 \leq n \leq 100$ )！

比如：n=3，那么 $s=2+2^2+2^2\cdot 2^2=14$ ！

### 输入

从键盘读入一个整数n ( $1 \leq n \leq 100$ )

### 输出

输出求出的和

### 样例输入

3

### 样例输出

14

a整数数组，存放2的n次方

6	1							
0	1	2	3	4	5	6	7	8

r整数数组，存放总和

0	3							
0	1	2	3	4	5	6	7	8

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a[100]={1}, r[1000]={0}, i, j, k=1, n, k2=1, len;
    //k代表a数组元素的个数，代表2的n次方高精度的整数的数位
    //k2代表了高精度的总和的位数
    cin>>n;
    //循环n次，每次都将a数组*2
    for(i=1;i<=n;i++){
        //将a数组的每一位都*2
        for(j=0;j<k;j++){
            a[j]=a[j]*2;
        }
        //逐位进位
        for(j=0;j<k;j++){
            if(a[j]>=10){
                a[j+1]=a[j+1]+a[j]/10;
                a[j]=a[j]%10;
            }
        }
        //判断a数组是否多一位
        if(a[k] != 0){
            k++;
        }
    }
}
```

```

//求出了2的i次方，结果为k位
//将k位的2的i次方加到k2位的总和r上
len=k;
if(k2>k) len = k2;
for(j=0;j<len;j++){
    r[j]=r[j]+a[j];
    //进位
    if(r[j]>=10){
        r[j+1]=r[j+1]+r[j]/10;
        r[j]=r[j]%10;
    }
}
//判断r数组是否多了1位
if(r[k2]!=0) k2++;
}
//输出r数组的结果
for(i=k2-1;i>=0;i--){
    cout<<r[i];
}
return 0;
}

```

## 2.2 课后作业

### 1. 【基础】计算N的阶乘

#### 题目描述

请计算n的阶乘 (1<=n<=100)

n的阶乘计算公式为： n!=n\*(n-1)\*(n-2)\*...\*1, 如： 5!=5\*4\*3\*2\*1=120

#### 输入

一个整数n(1<=n<=100)

#### 输出

n的阶乘

#### 样例输入

20

#### 样例输出

2432902008176640000

#### 参考题解：

```

#include<bits/stdc++.h>
using namespace std;
int a[10001]={1},i,n,j,len=1;
int main(){
    cin>>n;
}

```

```

for(i=1;i<=n;i++){
    for(j=0;j<len;j++){
        a[j]=a[j]*i;
    }
    len=len+2;
    for(j=0;j<len;j++){
        if(a[j]>=10){
            a[j+1]=a[j+1]+a[j]/10;
            a[j]=a[j]%10;
        }
    }
    while(a[len-1]==0){
        len--;
    }
}
for(i=len;i>=0;i--){
    if(a[i]!=0){
        len=i;
        break;
    }
}
for(i=len;i>=0;i--){
    cout<<a[i];
}
return 0;
}

```

## 2. 【基础】求 $1!+2!+3!+4!+\dots+n!$

### 题目描述

请求出 $1!+2!+3!+4!+\dots+n!$ , 请注意,  $n \leq 50$ 。

$n! = n * (n-1) * (n-2) \dots 1$ , 如:  $5! = 5 * 4 * 3 * 2 * 1 = 120$ 。

### 输入

请输入一个整数 $n$  ( $n \leq 50$ )

### 输出

输出求和的结果

### 样例输入

10

### 样例输出

4037913

### 示例题解

```

#include <bits/stdc++.h>
using namespace std;
int a[1000] = {1}; // 表示阶乘
int r[1000]; // 表示总和
int k, len; // k表示a数组下标, len表示r数组下标
int n;

```

```

int main() {
    cin>>n;
    for(int i = 1;i <= n;i++){
        for(int j = 0;j < k;j++){
            a[j] = a[j] * i;
        }
        //进位
        for(int j = 0;j < k + 2;j++){
            if(a[j] >= 10){
                a[j+1] = a[j+1] + a[j] / 10;
                a[j] = a[j] % 10;
            }
        }
        //判断结果是否多2位或者1位
        if(a[k+1] != 0) k = k + 2;
        else if(a[k] != 0) k++;
        //求和
        len = max(k,len);
        for(int j = 0;j < len;j++){
            r[j] = r[j] + a[j];
        }
        //进位
        for(int j = 0;j < len;j++){
            if(r[j] >= 10){
                r[j+1] = r[j+1] + r[j] / 10;
                r[j] = r[j] % 10;
            }
        }
        if(r[len] != 0) len++;
    }
    //输出结果
    for(int i = len - 1;i >= 0;i--){
        cout<<r[i];
    }
    return 0;
}

```

### 3. 【基础】棋盘里的麦子？

#### 题目描述

传说西塔发明了国际象棋而使国王十分高兴，他决定要重赏西塔，西塔说：“我不要你的重赏，陛下，只要你在我的棋盘上赏一些麦子就行了。在棋盘的第一个格子里放1粒，在第二个格子里放2粒，在第三个格子里放4粒，在第四个格子里放8粒，依此类推，以后每一个格子里放的麦粒数都是前一个格子里放的麦粒数的2倍，直到放满第64个格子就行了”。“区区小数，几粒麦子，这有何难，来人”，国王令人如数付给西塔。

计数麦粒的工作开始了，第一格内放1粒，第二格内放2粒第三格内放4粒，...还没有到第二十格，一袋麦子已经空了。一袋又一袋的麦子被扛到国王面前来。但是，麦粒数一格接一格飞快增长着，国王很快就看出，即便拿出全国的粮食，也兑现不了他对西塔的诺言。

请你编程帮助国王计算出，第n个棋盘格子中需要放多少粒麦子？

#### 输入

一个整数N代表第n格棋盘 ( $n \leq 100$ )

#### 输出

一个整数，代表第n格棋盘中麦子的总数。

## 样例输入

```
3
```

## 样例输出

```
4
```

## 示例代码

```
#include<bits/stdc++.h>
using namespace std;
string mul(string s){
    string r;
    int a[200]={0}, i, len;
    len=s.size();
    for(i=0; i<len; i++){
        a[i]=s[len-1-i]-'0';
    }
    for(i=0; i<len; i++){
        a[i]=a[i]*2;
    }
    for(i=0; i<len; i++){
        if(a[i]>=10){
            a[i+1]=a[i+1]+a[i]/10;
            a[i]=a[i]%10;
        }
    }
    if(a[len]!=0){
        len++;
    }
    for(i=len-1; i>=0; i--){
        r=r+to_string(a[i]);
    }
    return r;
}

int main(){
    int n, i;
    cin>>n;
    string s="1";
    for(i=1; i<n; i++){
        s=mul(s);
    }
    cout<<s;
    return 0;
}
```

## 4. 【提高】Pell数列

### 题目描述

有一种数列，它的前10项的值分别为：1 2 5 12 29 70 169 408 985 2378，这个数列被称为Pell数列，请问该数列的第n项的值是多少？(n<=1000)

### 输入

一个整数n

输出

第n项的值

样例输入

```
10
```

样例输出

```
2378
```

## 2.3 附加题

### 1. 【提高】小X与神牛

题目描述

神牛都长着B只角，B只角从左到右在头顶上排成一排。每只角上都标着数字，不是0就是1。小X将每头神牛的B只角上的数字从左到右依次取出，组成一个只含0或1的B位二进制数。小X将这个二进制数转化为十进制，用这个十进制数来代表一头神牛，这个十进制就是这头神牛的编号。

神牛们之间的关系是很微妙的，如果两头神牛的第*i*只角上的数字不同，则称这两头神牛的第*i*只角是不一样的。如果两头神牛不同的角的数目大于等于D，则称这两头神牛是友好的。比如当B=8，D=2时，

```
01010100
```

```
00110100
```

```
xx
```

这两头神牛的第2和第3只角不同(x指向的位置)，不同的角的数目为2，所以这两头神牛是友好的。

现在小X向你求助：请找出N头神牛，使得任意两头神牛都是友好的，并将这N头神牛的编号按从小到大排序后依次输出。如果有多种符合条件的解，那么排在越前面的牛的编号越小越好。

输入

输入仅有一行包含3个用空格隔开的正整数，分别表示 N, B, D。

输出

输出仅有一行包含N个非负整数，相邻两个数之间用一个空格隔开，表示N头神牛的编号。如果有多个解，你的程序要输出这样的解：越前面的牛的编号越小越好。

样例输入

```
3 5 3
```

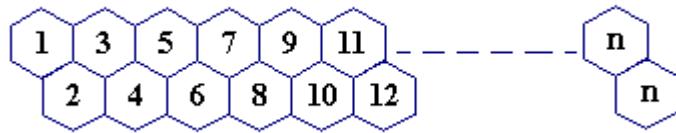
样例输出

```
0 7 25
```

### 2. 【提高】蜜蜂路线

题目描述

一只蜜蜂在下图所示的数字蜂房上爬动，已知它只能从标号小的蜂房爬到标号大的相邻蜂房，现在问你：蜜蜂从蜂房M开始爬到蜂房N， $1 \leq M < N \leq 100$ ，有多少种爬行路线？



### 输入

输入M, N的值。 (1<=m<=n<=100)

### 输出

爬行有多少种路线。

### 样例输入

```
1 14
```

### 样例输出

```
377
```

## 第二十六章 递推算法

### 1. 递推的基础知识

#### 1.1 什么是递推？

递推算法是一种用若干可重复运算描述复杂问题的方法，递推法是一种重要的数学方法，也是编程中解决问题的一个重要方法。

这种算法的特点是：一个问题的求解过程需要一系列的计算，在已知条件和所求问题之间总存在着某种相互关联关系，在计算时，如果可以找到前后过程之间的数量关系（即递推式），那么，从问题出发逐步推到已知条件。

无论顺推还是逆推，**关键是要找到递推式**。这种处理问题的方法能使复杂的运算化为若干重复的简单运算，充分发挥计算机擅长的重复处理能力。

递推算法的首要问题是找到相邻的数据项之间的关系（即递推关系）。递推算法避开了求通项公式的麻烦，把一个复杂的问题的求解分解成连续的若干步骤简单运算。一般来说，可以将递推算法看成一种特殊的迭代算法。

### 1.2 递推问题训练

#### 简单数值型递推

##### 1. 统计每个月兔子的总数

###### 题目描述

有一对兔子，从出生后第3个月起每个月都生一对兔子，一对小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问第n个月 ( $n \leq 50$ ) 的兔子总数为多少对？

###### 输入

输入1个整数n，表示第几个月

###### 输出

第n个月兔子的总数量有多少？

## 样例输入

9

## 样例输出

34

### 思路:

第1个月: a

第2个月: a

第3个月: a b

第4个月: a b c

第5个月: a b c d e

### 规律是什么?

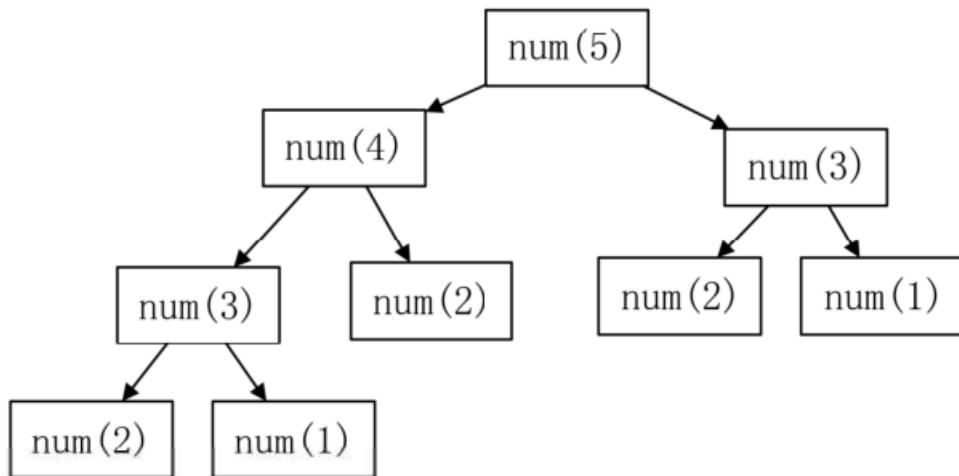
$a(n)=a(n-1)+a(n-2)$  (斐波那契数列)

1. 注意: 本题有两个起始项

```
#include<bits/stdc++.h>
using namespace std;
int num(int n){
    int r;
    if(n==1||n==2){
        r=1;
    }else{
        r=num(n-1)+num(n-2);
    }
    return r;
}
int main(){
    int n;
    cin>>n;
    cout<<num(n)<<endl;
    return 0;
}
```

### 补充:

1. 当n的值越来越大时, 求解的时间也会越来越差, 可能会引起超时, 因为随着n的值越来越大, 重复求解的项会越来越多, 下图的结构会非常庞杂:



2. int类型的数据最多只能表示 $2^{32}-1$ ,10位的整数,如果超出这个范围就需要使用long long类型来表达,而long long也有表达的范围极限: $2^{63}-1$ ,约20位的整数

### 递归存在的问题及解决方式

#### 1. 递归存在的问题

由于递归是函数对自己的调用,因此在n的值比较大时,存在太多重复求解的过程,而这些重复求解的过程必须要等到所有项计算结束才能一层层向上汇总得出结果,因此效率非常低。

#### 2. 提高递归效率的方法

##### 1. 数组

采用数组来记录每个月兔子的数量,只需要按需求计算一次即可得到所有数据,在求解第n个月的兔子的数量的时候,a[n-1]和a[n-2]就无需重复求解了

1	1	2	3	5		
---	---	---	---	---	--	--

#### 示例代码

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int n;
    long long a[51];
    cin>>n;
    a[0]=a[1]=1;
    for(int i=2;i<n;i++){
        a[i]=a[i-1]+a[i-2];
    }
    cout<<a[n-1]<<endl;
    return 0;
}

```

#### 2. 递推

我们用x和y分别标记第n-1和第n-2个月的兔子数量

1	1	2 x	3 y	5 n				
---	---	-----	-----	-----	--	--	--	--

```

#include<bits/stdc++.h>
using namespace std;
int main(){

```

```

long long x,y,n,s;
cin>>n;
x=1;
y=1;
for(int i=3;i<=n;i++){
    s=x+y;
    x=y;
    y=s;
}
if(n==1||n==2){
    cout<<1<<endl;
}else{
    cout<<s<<endl;
}

```

## 2. 猴子吃桃子

### 题目描述

猴子吃桃子问题：猴子第一天摘下若干个桃子，当即吃了一半还不过瘾，又多吃了两个；第二天又将剩下的桃子吃掉一半又多吃了两个；以后每天早上都吃了前一天剩下的一半零一个。到了第十天想再吃时，见只剩下一个桃子，求第一天共摘了多少个桃子？

### 输入

无

### 输出

一个整数，第一天共有多少个桃子

### 思路：

采用逆推法求解

```

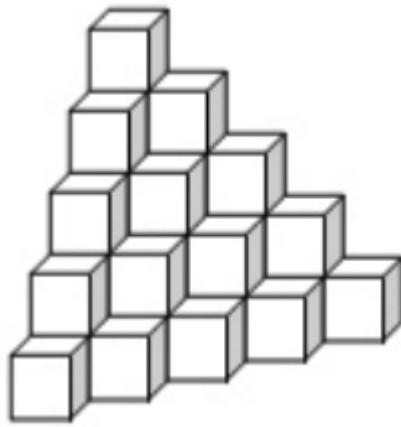
#include<bits/stdc++.h>
using namespace std;
int main(){
    int x,i;
    x=1;//第十天桃子数
    //逆推
    for(i=2;i<=10;i++){
        x=(x+1)*2;
    }
    cout<<x;
    return 0;
}

```

## 3. 数数小木块

### 题目描述

在墙角堆放着一堆完全相同的正方体小木块，如下图所示：



因为木块堆得实在是太有规律了，你只要知道它的层数就可以计算所有木块的数量了。

### 输入

只有一个整数  $n$ ，表示这堆小木块的层数，已知  $1 \leq n \leq 100$ 。

### 输出

只有一个整数，表示这堆小木块的总数量。

### 样例输入

```
5
```

### 样例输出

```
35
```

### 思路：

递推得出第*i*层小木块的数量再求和

```
#include<bits/stdc++.h>
using namespace std;
int i,x,s,n;//x表示第i层木块的数量
int main(){
    cin>>n;
    //An=An-1+n;
    //Ai=Ai-1+i
    //x=x+n;
    x=1;//第1层
    s=1;//第一层共有多少个
    //从第2层开始递推
    for(i=2;i<=n;i++){
        x=x+i;
        cout<<x<<endl;
        s+=x;
    }
    cout<<s;
    return 0;
}
```

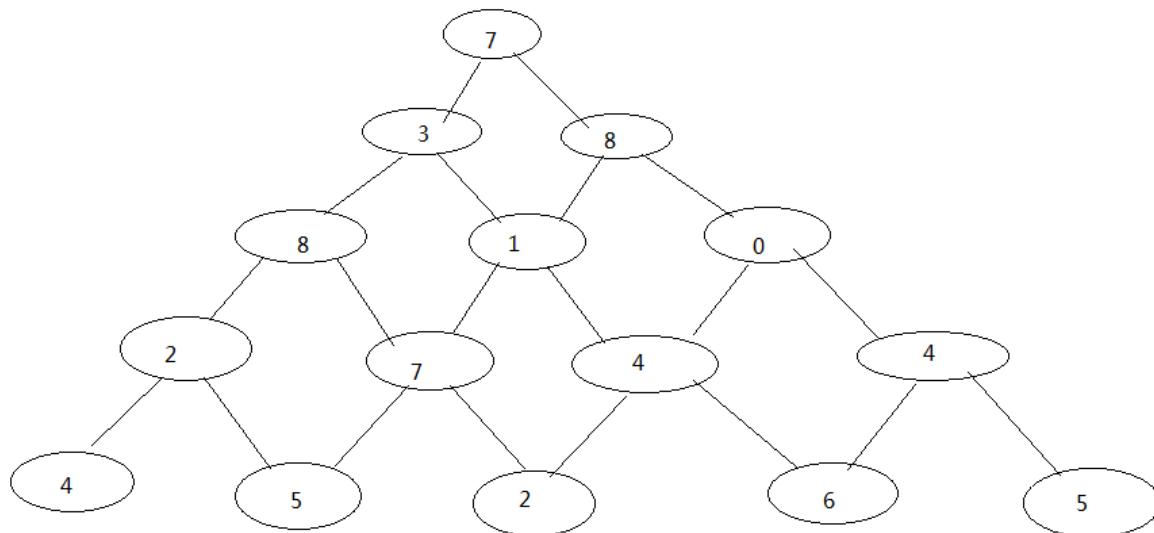
## 2. 递推问题应用

### 2.1 课堂案例

#### 1. 【基础】数塔问题

##### 题目描述

有如下所示的数塔，要求从底层走到顶层，若每一步只能走到相邻的节点，则经过的节点的数字之和最大是多少？



##### 输入

输入数据首先包括一个整数整数N( $1 \leq N \leq 100$ )，表示数塔的高度，接下来用N行数字表示数塔，其中第*i*行有个*i*个整数，且所有的整数均在区间[0,99]内。

##### 输出

从底层走到顶层经过的数字的最大和是多少？

##### 样例输入

```
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

##### 样例输出

```
30
```

##### 思路：

将数塔存入二维数组，从倒数第二层开始，递推计算出走到每个节点最多能累计的最大数字和是多少，直到第1层

层号							层号	第1层累加上， $a[i+1][j]$ 和 $a[i+1][j+1]$ 中较大的数				
1	7						1	30				
2	3	8					2	23	21			
3	8	1	0				3	20	13	10		
4	2	7	4	4			4	7	12	10	10	
5	4	5	2	6	5		5	4	5	2	6	5

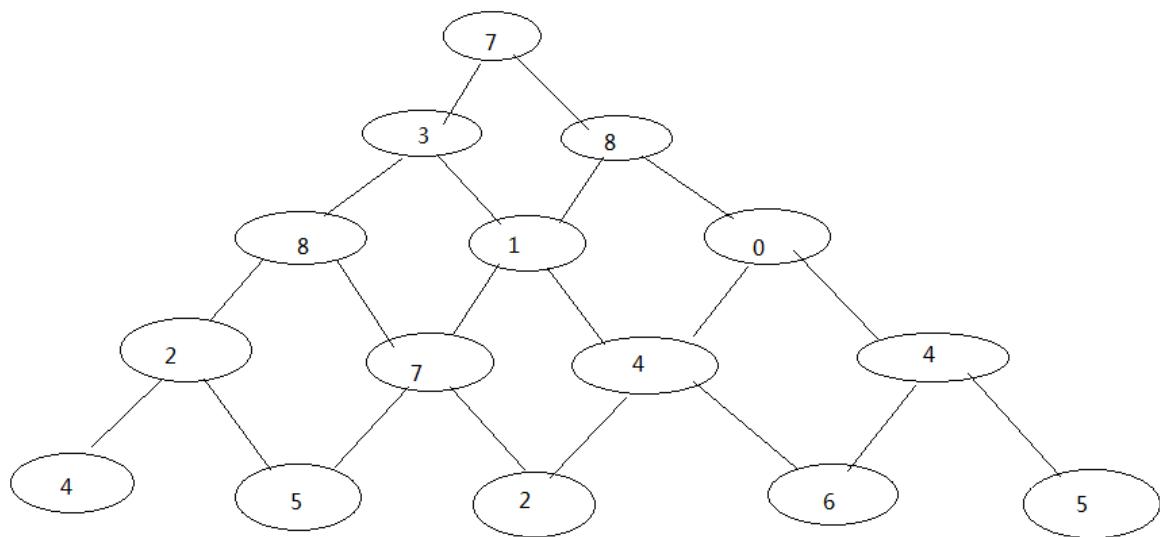
```
#include<bits/stdc++.h>
using namespace std;
int a[100][100];//存储数塔，以及走到每个点的最大和的结果
int r[100][3];//存储路线
int i,j,n,k;//k表示r数组的下标
int main(){
    cin>>n;
    //读入数塔
    for(i=1;i<=n;i++){
        for(j=1;j<=i;j++){
            cin>>a[i][j];
        }
    }
    //从倒数第2层开始逆推， 每个点累加下方的值和下方右边的值中较大的数
    for(i=n-1;i>=1;i--){
        for(j=1;j<=i;j++){
            if(a[i+1][j]>a[i+1][j+1]){
                a[i][j]=a[i][j]+a[i+1][j];
            }else{
                a[i][j]=a[i][j]+a[i+1][j+1];
            }
        }
    }
    cout<<a[1][1];
    return 0;
}
```

## 2. 【基础】数塔的行走路径?

### 题目描述

有如下所示的数塔，要求从底层走到顶层，若每一步只能走到相邻的节点，要求经过节点的数字之和最大，请问应该如何走，请输出从塔底到塔顶的行走路线，同时计算出经过节点的最大数字和是多少？

(假设本问题中，不存在多条路线从塔底走到塔顶经过节点的数字和都是最大的，也就是本题涉及测试数据得到的路径都是唯一的)



为了方便计算我们将数塔中的值存到如下图所示的二维数组中，该数塔的行走路线将如下图的箭头所示。

	1	2	3	4	5
1	7				
2	3	8			
3	8	1	0		
4	2	7	4	4	
5	4	5	2	6	5

## 输入

输入数据首先包括一个整数整数N( $1 \leq N \leq 100$ )，表示数塔的高度，接下来用N行数字表示数塔，其中第i行有个i个整数，且所有的整数均在区间[0,99]内。

## 输出

第一行，按照样例输出所示的形式，输出数塔的行走路线。

第二行，输出经过节点的最大数字和。

## 样例输入

```
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

## 样例输出

5, 2->4, 2->3, 1->2, 1->1, 1

30

## 思路：

第一步将数塔存入二维数组，从倒数第二层开始，递推计算出走到每个点最多能够累计的最大数字和是多少，直到第1层

第二步倒过来推导一下每一个数都是从哪个点过来的，记录下对应的点的值再次逆序输出

层号						层号	第i层累加上，a[i+1][j]和a[i+1][j+1]中较大的数					r数组：存储路径		
1	7					1	30					1	1	1
2	3	8				2	23	21				k	2	2
3	8	1	0			3	20	13	10			3	3	1
4	2	7	4	4		4	7	12	10	10		4	4	2
5	4	5	2	6	5	5	4	5	2	6	5	5	5	2

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
/*
```

思路：第一步：采用逆推法，计算出走到每个点经过的数字和的最大值

第二步：从第一层开始逐层向下计算（递推），算出每个点是从下方来的还是从下方右侧来的，将路线记录到数组中

```
*/
```

```
int a[100][100];//存储数塔，以及走到每个点的最大和的结果
```

```
int r[100][3];//存储路线
```

```
int i,j,n,k,s;//k表示r数组的下标
```

```
int main(){
```

```
    cin>>n;
```

```
    //读入数塔
```

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

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

```
            cin>>a[i][j];
```

```
        }
```

```
}
```

```
//计算走到每个点经过的数字最大和
```

```
    for(i=n-1;i>=1;i--){
```

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

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


```

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

```

```
            }else{

```

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

```

```
            }

```

```
        }
```

```
}
```

```
s=a[1][1];
```

```
// cout<<s;
```

```
//计算路径
```

```
//第一个点即终点，无需计算
```

```
k=1;
```

```
r[1][1]=1;
```

```
r[1][2]=1;
```

```
//表示正在计算的点
```

```

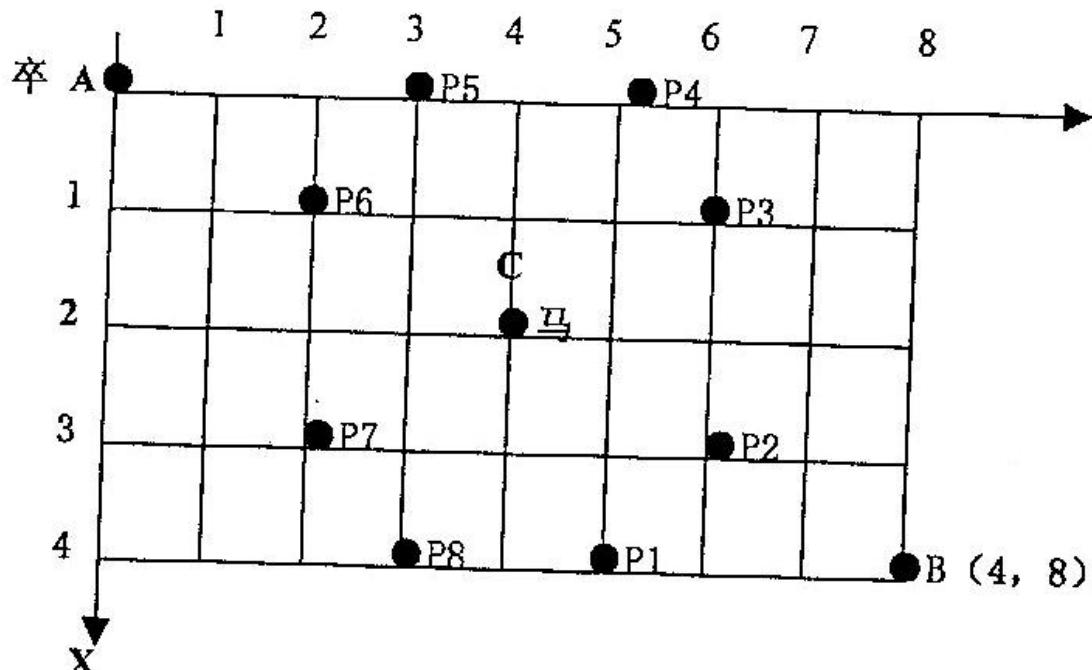
i=1;
j=1;
//推导第2行的点是从哪个点上来的，直到第n行结束
while(i<n){
    k++;
    if(a[i+1][j]>a[i+1][j+1]){
        r[k][1]=i+1;
        r[k][2]=j;
        i++;
    }else{
        r[k][1]=i+1;
        r[k][2]=j+1;
        i++;
        j++;
    }
}
//逆序从底层到顶层打印路线
for(i=k;i>=1;i--){
    cout<<r[i][1]<<","<<r[i][2];
    if(i!=1){
        cout<<"->";
    }
}
cout<<endl<<s;
return 0;
}

```

### 3. 【提高】过河卒

#### 题目描述

A点有一个过河卒，需要走到目标B点。卒行走规则：可以向下、或者向右。同时在棋盘上的任一点有一个对方的马（如下图的C点），该马所在的点和所有跳跃一步可达的点称为对方马的控制点。例如下图C点可以控制9个点（图中的P1, P2 ... P8 和 C）。卒不能通过对方马的控制点。棋盘用坐标表示，现给定A点位置为(0,0) B点位置为(n,m) (n,m为不超过20的整数)，马的位置为C(X,Y)（约定：C点与A点不重叠，与B点也不重叠）。要求你计算出卒从A点能够到达B点的路径的条数。



## 输入

B点的坐标 (n,m) 以及对方马的坐标 (X,Y) (马的坐标一定在棋盘范围内，但要注意，可能落在边界的轴上)

## 输出

### 样例输入

```
6 6 3 2
```

### 样例输出

```
17
```

### 思路：

第1行和第1列的走法是确定的，因为题目要求只能向右或者向下走；

其余的点走到 $a[i][j]$ 点的路径条数= $a[i-1][j]+a[i][j-1]$

	0	1	2
0	1	1	1
1	1	2	3
2	1	3	6
3	1	4	10

	0	1	2	3	4	5	6
0	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0
2	1	1	1	马	0	0	0
3	1	0	1	0	0	0	0
4	1	1	0	0	0	0	0
5	1	2	2	2	2	2	2
6	1	3	5	7	9	11	13

将棋盘所有值设为1，标记马及马的控制点为0.

	0	1	2	3	4	5	6
0	1	1	0	1	0	1	1
1	1	0	1	1	1	0	1
2	1	1	1	马	1	1	1
3	1	0	1	1	1	0	1
4	1	1	0	1	0	1	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1

a[0][0]无需计算，除此以外：

i=0,第1行，除了控制点，a[i][j]等于左侧的值

j=0,第1列，除了控制点，a[i][j]等于上侧的值

其余的点，除了控制点，a[i][j]等于左侧的值加上侧的值

	0	1	2	3	4	5	6
0	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0
2	1	1	1	马	0	0	0
3	1	0	1	0	0	0	0
4	1	1	0	0	0	0	0
5	1	2	2	2	2	2	2
6	1	3	5	7	9	11	13

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
/*
将棋盘所有值都设置为1
标记马及马的控制点为0
a[0][0]无需计算，除此以外，
i=0,第1行，除了控制点，a[i][j]等于左侧的值
j=0,第1列，除了控制点，a[i][j]等于上侧的值
其余的点，除了控制点，a[i][j]等于左侧的值加上侧的值
*/
int main(){
    int a[7][7];
    for(int i=0;i<7;i++)
        for(int j=0;j<7;j++)
            a[i][j]=1;
    a[2][2]=0;
    a[3][5]=0;
    a[5][3]=0;
    a[0][0]=1;
    for(int i=1;i<7;i++)
        for(int j=1;j<7;j++)
            if(i==j)
                a[i][j]=a[i-1][j]+a[i][j-1];
            else if(i==j+1)
                a[i][j]=a[i-1][j];
            else if(j==i+1)
                a[i][j]=a[i][j-1];
            else
                a[i][j]=a[i-1][j]+a[i][j-1];
    for(int i=0;i<7;i++)
        for(int j=0;j<7;j++)
            cout<<a[i][j]<<" ";
    return 0;
}
```

```

int a[30][30];//存放棋盘的值， 默认初始值全为0
int n,m,x,y,i,j;
cin>>n>>m>>x>>y;
//初始化所有值为1
for(i=0;i<=n;i++){
    for(j=0;j<=m;j++){
        a[i][j]=1;
    }
}
//设置马及马的控制点为0
a[x][y]=0;
//如果控制点在棋盘内， 就设为0
if(x-1>=0&&y-2>=0) a[x-1][y-2]=0;
if(x-2>=0&&y-1>=0) a[x-2][y-1]=0;
if(x-2>=0&&y+1<=m) a[x-2][y+1]=0;
if(x-1>=0&&y+2<=m) a[x-1][y+2]=0;
if(x+1<=n&&y+2<=m) a[x+1][y+2]=0;
if(x+2<=n&&y+1<=m) a[x+2][y+1]=0;
if(x+1<=n&&y-2<=m) a[x+1][y-2]=0;
if(x+2<=n&&y-1>=0) a[x+2][y-1]=0;
/*for(i=0;i<=n;i++){
    for(j=0;j<=m;j++){
        cout<<a[i][j]<<" ";
    }
    cout<<endl;
}*/
//递推计算
for(i=0;i<=n;i++){
    for(j=0;j<=m;j++){
        //起点、 控制点不计算
        if(i==0&&j==0||a[i][j]==0){
            continue;
        }
        //如果是第1行
        if(i==0){
            a[i][j]=a[i][j-1];
        }else if(j==0){
            a[i][j]=a[i-1][j];
        }else{
            a[i][j]=a[i-1][j]+a[i][j-1];
        }
    }
    cout<<a[n][m];
    return 0;
}

```

#### 4. 【提高】Pell数列

##### 题目描述

有一种数列，它的前10项的值分别为：1 2 5 12 29 70 169 408 985 2378，这个数列被称为Pell数列，请问该数列的第n项的值是多少？(n<=1000)

##### 输入

一个整数n

输出

第n项的值

样例输入

10

样例输出

2378

思路：

1	2	5	12	29	70	169	408	985	2378
1	2	3	4	5	6	7	8	9	10

观察可得规律为：

$$A_n = A_{n-1} * 2 + A_{n-2}$$

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
/*An=A(n-1)*2+ =A(n-2)*/
//高精度求两数的和
string sum(string s1,string s2){
    string r;//结果
    int i, a[1000] = {0}, b[1000] = {0}, c[1000] = {0};
    //逆序存入数组
    for(i=0;i<s1.size();i++){
        a[i]=s1[s1.size()-i-1]-'0';
    }
    for(i=0;i<s2.size();i++){
        b[i]=s2[s2.size()-i-1]-'0';
    }
    int len=s1.size();
    if(len<s2.size()) len=s2.size();
    //逐位相加，逐位进位
    for(i = 0;i<len;i++){
        c[i]=c[i]+a[i]+b[i];
        if(c[i]>=10){
            c[i+1]=c[i+1]+c[i]/10;
            c[i]=c[i]%10;
        }
    }
    //判断是否多一位
    if(c[len] !=0) len++;
    //逆序输出
    for(i=len-1;i>=0;i--){
        r=r+(char)(c[i]+'0');
    }
}
```

```

        return r;
    }
    //高精度数*2的值
    string mul(string s){
        string r;
        int a[1000]={0},i;
        //逆序存入a数组
        for(i=0;i<s.size();i++){
            a[i]=s[s.size()-1-i]-'0';
        }
        //逐位*2
        for(i=0;i<s.size();i++){
            a[i]=a[i]*2;
        }
        //逐位进位
        for(i=0;i<s.size();i++){
            if(a[i]>=10){
                a[i+1]=a[i+1]+a[i]/10;
                a[i]=a[i]%10;
            }
        }
        //判断是否多一位
        int len=s.size();
        if(a[len]!=0) len++;
        //逆序输出
        for(i=len-1;i>=0;i--){
            r=r+to_string(a[i]);
        }
        return r;
    }
    int main(){
        //z:代表计算结果，xy表示z的前两项
        string x,y,z;
        int i,n;
        // cout<<mul("12345");
        // cout<<sum("123","321");
        cin>>n;
        /*An=A(n-1)*2+ =A(n-2)*/
        x="1";
        y="2";
        if(n==1){
            cout<<x;
        }else if(n==2){
            cout<<y;
        }else{
            //从第3项开始递推
            for(i=3;i<=n;i++){
                z=sum(mul(y),x);
                //修改x,y的值
                x=y;
                y=z;
            }
        }
        cout<<z;
        return 0;
    }
}

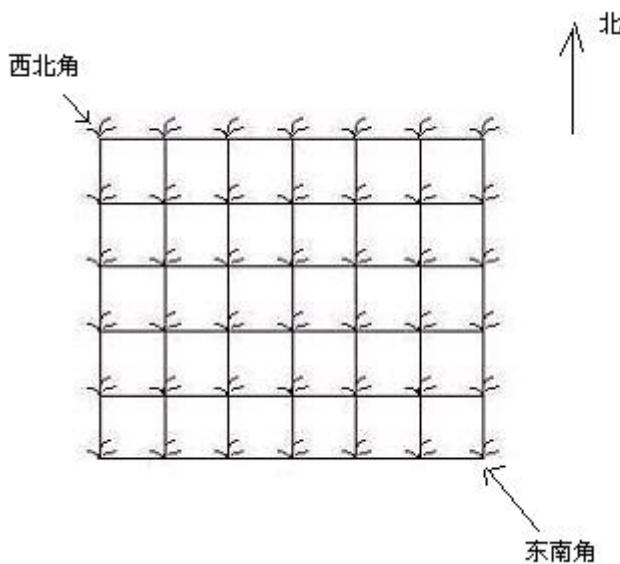
```

## 2.2 课后作业

### 1. 【基础】摘花生问题

#### 题目描述

Hello Kitty 想摘点花生送给她喜欢的米老鼠。她来到一片有网格状道路的矩形花生地(如下图)，从西北角进去，东南角出来。地里每个道路的交叉点上都有种着一株花生苗，上面有若干颗花生，经过一株花生苗就能摘走该它上面所有的花生。Hello Kitty只能向东或向南走，不能向西或向北走。问Hello Kitty 最多能够摘到多少颗花生。



如输入：

2 2

1 1

3 4

代表有2行，每行有2株花生，那么摘能摘到的最多的花生就是：1->3->4，总和为8颗花生。

#### 输入

第一行是两个整数m和n ( $m \leq 100, n \leq 100$ )，代表了花生地里有m行，每行有n列的花生！

后面m行，每行有n个整数代表了每行中，每株花生的数量！

#### 输出

输出是一个整数，代表了最多能摘到的花生的总数

#### 样例输入

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

#### 样例输出

```
8
```

### 2. 【基础】摘花生问题（2）

## 题目描述

Hello Kitty又一次来到花生地里摘花生，从左上角进入花生地，从右下角出去，只能向右或者向下，请问Hello Kitty应该沿着什么样的路线走，能够摘到的花生数量最多（假设花生地里没有任何2株的花生一样多，也不存在多条路线能够摘到一样多的花生的情况）？

比如输入：

2 2

1 2

3 4

应该输出：1-3-4，也就是按照1 3 4这三株数量的花生摘过去，能够摘到最多的花生！

## 输入

第一行是2个整数m和n ( $2 \leq m, n \leq 100$ )，代表花生地有m行，n列花生！

后面m行，每行有n个整数代表了每行中，每株花生的数量

## 输出

输出Hello Kitty按照走过的路线中，摘到每株花生的数量。

## 样例输入

2 2

1 2

3 4

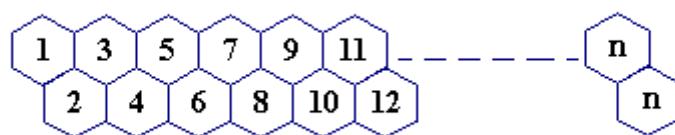
## 样例输出

1-3-4

## 3. 【提高】蜜蜂路线

### 题目描述

一只蜜蜂在下图所示的数字蜂房上爬动，已知它只能从标号小的蜂房爬到标号大的相邻蜂房，现在问你：蜜蜂从蜂房M开始爬到蜂房N， $1 \leq M < N \leq 100$ ，有多少种爬行路线？



## 输入

输入M，N的值。 ( $1 \leq M < N \leq 100$ )

## 输出

爬行有多少种路线。

## 样例输入

1 14

## 样例输出

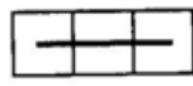
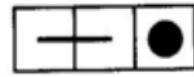
377

## 4. 【入门】骨牌铺方格

### 题目描述

有 $1 \times n$  ( $n \leq 50$ ) 的一个长方形，用一个 $1 \times 1$ 、 $1 \times 2$ 和 $1 \times 3$ 的骨牌铺满方格，请问有多少种铺法？

例如当 $n=3$ 时为 $1 \times 3$ 的方格。此时用 $1 \times 1$ 、 $1 \times 2$ 和 $1 \times 3$ 的骨牌铺满方格，共有四种铺法。如下图：



### 输入

一个整数 $n$  ( $n \leq 50$ )

### 输出

骨牌的铺法

### 样例输入

```
3
```

### 样例输出

```
4
```

## 5. 【入门】小X放骨牌

### 题目描述

小X喜欢下棋。

这天，小X对着一个长为 $N$ 宽为 $M$ 的矩形棋盘发呆，突然想到棋盘上不仅可以放棋子，还可以放多米诺骨牌。

每个骨牌都是一个长为2宽为1的矩形，当然可以任意旋转。小X想知道在骨牌两两不重叠的前提下，这个棋盘上最多能放多少个骨牌，希望你帮帮他。

### 输入

第一行包含用一个空格隔开的两个整数 $N, M$ 。

### 输出

第一行包含一个整数，表示该棋盘上最多能放的骨牌个数。

### 样例输入

```
2 3
```

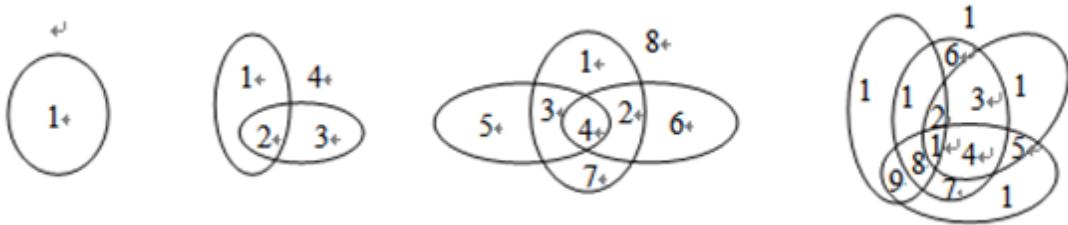
### 样例输出

```
3
```

## 6. 【入门】平面分割问题

### 题目描述

设有 $n$ 条封闭曲线画在平面上，而任何两条封闭曲线恰好相交于两点，且任何三条封闭曲线不相交于同一点，问这些封闭曲线把平面分割成的区域个数。



### 输入

一个整数n ( $n \leq 10000$ )，代表封闭曲线的条数

### 输出

n条曲线分割区域的个数

### 样例输入

2

### 样例输出

4

## 第二十七章 贪心算法

### 1. 贪心算法的基础知识

#### 1.1 什么是贪心算法

贪心算法（又称“贪婪算法”）是指：在对问题求解时，总是做出当前看来是最好的选择。，也就是说不从整体最优解考虑，它所做出的是在某种意义上的局部最优解。

贪心算法不是对所有问题都能得到整体最优解，关键是贪心策略的选择。选择的贪心策略必须具备无后效性，即某个状态以前的过程不会影响以后的状态，只与当前的状态有关。

贪心算法的使用前提：**局部最优解一定能导致全局最优解。**

接触过的贪心问题：数塔，摘花生，过河卒。

**贪心的解决策略：**

1. 建立数学模型描述问题
2. 把求解的问题分成若干个子问题
3. 对每一个子问题求解，得到子问题的局部最优解
4. 把子问题的局部最优解合成原来问题的一个解

### 2. 贪心问题练习

#### 2.1 课堂案例

1. 【入门】需要安排几位师傅加工零件？

##### 题目描述

某工厂有n个零件加工的师傅，每位师傅每天能够加工出不同数量的零件。现有m个零件要求一天加工完，请问该工厂最少需要派几个师傅来完成这次零件加工任务，如果安排所有的师傅都参与加工也不能在一天内完成任务，请输出“NO”。

## 输入

第一行有两个整数，用空格隔开；第一个整数代表要加工的总零件个数m (m<=10^6)，第二个整数代表工厂的零件加工师傅的数量n (n<=100)。

第二行有n个整数，分别代表每个师傅每天能够加工出来的零件数量（每个师傅每天加工的零件数量<=10^4）。

## 输出

工厂在1天时间内加工所有零件需要的师傅数量或者输出NO。

### 样例输入

```
10 5
1 3 2 4 2
```

### 样例输出

```
4
```

## 思路

由于问题是需要最少多少个师傅，所以我们优先挑选加工能力强的师傅

### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int m,n,a[101]={0},i,j,t,s=0,c=0;
    cin>>m>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    for(i=0;i<n-1;i++){
        for(j=0;j<n-i-1;j++){
            if(a[j]<a[j+1]){
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
    i=0;
    while(s<m){
        c++;
        s+=a[i];
        i++;
    }
    if(!(i<n)){
        cout<<"NO";
    }else{
        cout<<c;
    }
    return 0;
}
```

**补充：**排序可以用sort和辅助排序函数来快速实现。

## 2. 【基础】排队打水问题

### 题目描述

有n个人排队到r个水龙头去打水，他们装满水桶的时间t<sub>1</sub>,t<sub>2</sub>,...,t<sub>n</sub>为整数且各不相等，应如何安排他们的打水顺序才能使他们花费的总时间最少？

每个人打水的时间 = 排队的时间 + 实际打水的时间，本题假设一个人打好水，排在他后面的人接着打水的这个切换过程不消耗时间。

比如，有2个人A和B，他们打水的时间分别是3和2，只有1个水龙头，这时，如果A先打水，B后打水，那么A和B打水的时间分别为3、3+2（B排队3分钟）。

因此，所有人打水的总时间就是每个人的打水时间及每个人的排队时间的总和。

### 输入

第1行，两个整数n(1<=n<=500)和r(1<=r<=100)。

第2行，n个正整数t<sub>1</sub>,t<sub>2</sub>,...,t<sub>n</sub>，(1<=t<sub>i</sub><=1000)表示每个人装满水桶的时间。

### 输出

1行，一个正整数，表示他们花费的最少总时间。

### 样例输入

```
4 2  
2 6 4 5
```

### 样例输出

```
23
```

### 思路

方案一			方案二			方案三					
龙头1	龙头2		龙头1	龙头2		龙头1	龙头2				
2	5		5	6		2	4				
4	6		2	4		5	6				
总计	8	16	24	总计	12	16	28	总计	9	14	23

每个人打水的时间=该用户的打水时间+排队等待的时间，因此只需要让打水快的人先打就可以减少总时间。

假设有6个人，打水时间分别为：2,5,8,6,4,9，最少打水时间：

打水时间：

a	2	4	5	6	8	9
	1	2	3	4	5	6

每个人的打水时间：

a	2	4	7	10	15	19
	1	2	3	4	5	6

两个水龙头，前r个人无需排队，从第r+1个人开始， $a[i]=a[i]+a[i-r]$

$a[3]=a[3]+a[1]$

$a[4]=a[4]+a[2]$

$a[5]=a[5]+a[3]$

$a[6]=a[6]+a[4]$

### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n,r,i,s=0,a[501];
    cin>>n>>r;
    for(i=1;i<=n;i++){
        cin>>a[i];
    }
    //排序
    sort(a+1,a+n+1);
    //计算每个人的打水时间
    for(i=1;i<=n;i++){
        //从第r+1个人开始计算
        if(i>=r+1){
            a[i]=a[i]+a[i-r];
        }
        s+=a[i];
    }
    cout<<s;
    return 0;
}
```

### 3. 【提高】拦截导弹的系统数量求解

#### 题目描述

某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。

假设某天雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。

输入n个导弹依次飞来的高度（给出的高度数据是不大于30000的正整数），计算如果要拦截所有导弹最少要配备多少套这种导弹拦截系统。

比如：有8颗导弹，飞来的高度分别为

389 207 175 300 299 170 158 165

那么需要2个系统来拦截，他们能够拦截的导弹最优解分别是：系统1：拦截 389 207 175 170  
158

系统2：拦截 300 299 165

#### 输入

两行，第一行表示飞来导弹的数量n ( $n \leq 1000$ )

第二行表示n颗依次飞来的导弹高度

#### 输出

要拦截所有导弹最小配备的系统数k

### 样例输入

```
8  
389 207 175 300 299 170 158 165
```

### 样例输出

```
2
```

### 思路

找当前拦截系统中拦截高度最矮的系统拦截当前的导弹。

导弹

389	207	175	300	299	182	160	165
-----	-----	-----	-----	-----	-----	-----	-----

拦截策略：

如果没有系统可以拦截，就新开一个系统，修改系统高度；

如果可以拦截，则找到当前拦截高度最矮的系统进行拦截，修改系统高度为当前导弹的高度。

特点：

由于采用上述策略，导致前面的拦截系统的高度<后面的系统

### 示例代码

```
#include<bits/stdc++.h>  
using namespace std;  
/*  
x表示每个导弹高度  
p为找到的能拦截导弹的系统下标  
k为a数组中已经有的能够拦截导弹的系统数量  
*/  
int a[1001], i, n, x, k, p, j;  
int main(){  
    cin >> n;  
    for(i=0; i < n; i++){  
        cin >> x;  
        p = -1;  
        //循环a数组，找到第一个能拦截的系统  
        for(j=1; j <= k; j++){  
            if(a[j] >= x){  
                p = j;  
                break;  
            }  
        }  
        //如果没有找到系统拦截  
        if(p == -1){  
            k++;  
            a[k] = x; //设置系统能拦截的最高高度  
        } else{  
            //用第p个系统拦截，修改系统的最高高度  
            a[p] = x;  
        }  
    }  
}
```

```
    cout<<k;
    return 0;
}
```

#### 4. 【基础】活动选择

##### 题目描述

学校在最近几天有n ( $n \leq 100$ ) 个活动，这些活动都需要使用学校的大礼堂，在同一时间，礼堂只能被一个活动使。由于有些活动时间上有冲突，学校办公室人员只好让一些活动放弃使用礼堂而使用其他教室。

现在给出n个活动使用礼堂的起始时间begini和结束时间endi( $begini < endi$ )，请你帮助办公室人员安排一些活动来使用礼堂，要求安排的活动尽量多。请问最多可以安排多少活动？

请注意，开始时间和结束时间均指的是某个小时的0分0秒，如：3 5，指的是3:00~5:00，因此3 5和5 9这两个时间段不算冲突的时间段。

##### 输入

第一行一个整数n( $n \leq 100$ )

接下来的n行，每行两个整数，第一个begini，第二个是endi( $begini < endi \leq 32767$ )

##### 输出

输出最多能安排的活动数

##### 样例输入

```
11
3 5
1 4
12 14
8 12
0 6
8 11
6 10
5 7
3 8
5 9
2 13
```

##### 样例输出

```
4
```

##### 思路：

按结束时间排序，下一个活动开始时间对得上就可以安排，以此类推。

```
#include<bits/stdc++.h>
using namespace std;
int n,b[101],e[101],i,j,c=1,f;
int main(){
    cin>>n;
    for(i=0;i<n;i++){
        cin>>b[i]>>e[i];
    }
```

```

    }
    for(i=0;i<n-1;i++){
        for(j=0;j<n-i-1;j++){
            if(e[j]>e[j+1]){
                swap(b[j],b[j+1]);
                swap(e[j],e[j+1]);
            }
        }
    }
/* cout<<"排序结果: \n";
for(i=0;i<n;i++){
    cout<<b[i]<<" "<<e[i]<<endl;
}*/
f=e[0];
for(i=1;i<n;i++){
    if(b[i]>=f){
        c++;
        f=e[i];
    }
}
cout<<c;
return 0;
}

```

## 2.3 课后作业

### 1. 【提高】拦截导弹方案求解

#### 题目描述

某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：

虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。

某天，雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。

输入n个导弹依次飞来的高度（雷达给出的高度数据是不大于30000的正整数），计算如果要拦截所有导弹最少要配备多少套这种导弹拦截系统。

比如：有8颗导弹，飞来的高度分别为

389 207 175 300 299 170 158 165

那么需要2个系统来拦截，他们能够拦截的导弹最优解分别是：

系统1：拦截 389 207 175 170 158

系统2：拦截 300 299 165

#### 输入

两行，第一行表示飞来导弹的数量n (n<=1000)

第二行表示n颗依次飞来的导弹高度（导弹高度互不相等）

#### 输出

第一行输出：要拦截所有导弹最小配备的系统数k

接下来k行分别输出每套系统拦截哪些高度的导弹

#### 样例输入

8

389 207 175 300 299 170 158 165

## 样例输出

```
2
1:389 207 175 170 158
2:300 299 165
```

## 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int a[1000][1000],n,x,i,j,k,p,y,q;
int main(){
    cin>>n;
    for(i=0;i<n;i++){
        cin>>x;
        p=-1;
        for(j=0;j<=k;j++){
            for(q=0;q<n;q++){
                if(a[j][q]>=x&&a[j][q+1]==0){
                    a[j][q+1]=x;
                    p=j;
                    break;
                }
            }
            if(p!==-1){
                break;
            }
        }
        if(p== -1){
            k++;
            a[k][0]=x;
        }
    }
    cout<<k<<endl;
    for(i=1;i<=k;i++){
        cout<<i<<" :";
        for(j=0;j<=n;j++){
            if(a[i][j] !=0)cout<<a[i][j]<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

## 2. 【基础】删数问题

### 题目描述

键盘输入一个高精度的正整数n (n<=1000位) , 去掉其中任意s个数字后剩下的数字按原左右顺序将组成一个新的正整数。编程对给定的n和s (s<n的位数, 且数据保证n删除s个数之后不为0, 还是一个非0的整数) , 寻找一种方案, 使得剩下的数字组成的数最小。

例如: 153748要删除2个数, 使得剩下的数字最小, 应当删除5和7, 得到1348。 (注意: 1087如果要删除1个数, 删除1结果是最小的, 得到结果87)

### 输入

第一行是一个高精度整数n

第二行是需要删除的位数s

### 输出

最后剩下的最小数

### 样例输入

```
153748  
2
```

### 样例输出

```
1348
```

### 示例代码:

```
#include<bits/stdc++.h>  
using namespace std;  
int i,s,j,len,sum,f,k;  
string n;  
int main(){  
    cin>>n>>s;  
    len=n.size();  
    for(i=0;i<s;i++){  
        for(j=0;j<len;j++){  
            if(n[j]>n[j+1]){  
                for(int k=j;k<len;k++){  
                    n[k]=n[k+1];  
                }  
                break;  
            }  
        }  
    }  
    for(i=0;i<len-s;i++){  
        if(n[i]!='0'){  
            k=i;  
            break;  
        }  
    }  
    for(i=k;i<len-s;i++){  
        cout<<n[i];  
    }  
    return 0;  
}
```

### 3. 【基础】均分纸牌

#### 题目描述

有n堆纸牌 ( $2 \leq n \leq 200$ )，排成一行，编号分别为1,2,...n。已知每堆纸牌有一定的张数，且张数之和均为n的倍数。移动各堆中的任意张纸牌，使每堆的数量达到相同，且移动次数最少。

移动规则：

每次可以移动任意的张数，第1堆可以移向第2堆，第2堆可以移向第1堆或第3堆。。。第n堆只可以移向第n-1堆。例如，当n=4时：

堆号	1	2	3	4
张数	3	5	4	8

移动的方法有许多种，其中的一种方案：

① 第2堆向第1堆移动2张，成为：5 3 4 8

② 第4堆向第3堆移动3张，成为：5 3 7 5

③ 第3堆向第2堆移动2张，成为：5 5 5 5

经过三次移动，每堆都成为5张。

#### 输入

第一行一个整数n。

第二行n个整数，用空格分隔。

#### 输出

一个整数（表示最少移动次数）。

#### 样例输入

```
4
3 5 4 8
```

#### 样例输出

```
3
```

#### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int a[201], i, n, s, avg, c=0;
int main(){
    cin>>n;
    for(i=1; i<=n; i++){
        cin>>a[i];
        s+=a[i];
    }
    avg=s/n;
    for(i=1; i<=n; i++){
        if(a[i]<avg){
            a[i+1]=a[i+1]-(avg-a[i]);
            c++;
        }else if(a[i]>avg){
            a[i]=a[i]-(avg-a[i]);
            a[i+1]=a[i+1]+(avg-a[i]);
            c++;
        }
    }
}
```

```
    }
    cout<<c;
    return 0;
}
```

#### 4. 【基础】接水问题

##### 题目描述

学校里有一个水房，水房里一共装有m个龙头可供同学们打开水，每个龙头每秒钟的供水量相等，均为1。

现在有n名同学准备接水，他们的初始接水顺序已经确定。将这些同学按接水顺序从1到n编号，i号同学的接水量为wi。接水开始时，1到m号同学各占一个水龙头，并同时打开水龙头接水。当其中某名同学j完成其接水量要求wj后，下一名排队等候接水的同学k马上接替j同学的位置开始接水。这个换人的过程是瞬间完成的，且没有任何水的浪费。即j同学第x秒结束时完成接水，则k同学第x+1秒立刻开始接水。若当前接水人数n'不足m，则只有n'个龙头供水，其它m-n'个龙头关闭。

现在给出n名同学的接水量，按照上述接水规则，问所有同学都接完水需要多少秒。（noip2010普及组复赛第2题）

输入样例1：

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

输出样例1：

```
4
```

输入样例2：

```
8 4
23 71 87 32 70 93 80 76
```

输出样例2：

```
163
```

输入样例1的说明：

第1秒，3人接水。第1秒结束时，1、2、3号同学每人的已接水量为1，3号同学接完水，4号同学接替3号同学开始接水。

第2秒，3人接水。第2秒结束时，1、2号同学每人的已接水量为2，4号同学的已接水量为1。

第3秒，3人接水。第3秒结束时，1、2号同学每人的已接水量为3，4号同学的已接水量为2。4号同学接完水，5号同学接替4号同学开始接水。

第4秒，3人接水。第4秒结束时，1、2号同学每人的已接水量为4，5号同学的已接水量为1。1、2、5号同学接完水，即所有人完成接水。

总接水时间为4秒。

数据范围：

$1 \leq n \leq 10000$ ,  $1 \leq m \leq 100$  且  $m \leq n$ ;  
 $1 \leq w_i \leq 100$ 。

##### 输入

第1行2个整数n和m，用一个空格隔开，分别表示接水人数和龙头个数。

第2行n个整数w1、w2、.....、wn，每两个整数之间用一个空格隔开，wi表示i号同学的接水量。

##### 输出

输出只有一行，1个整数，表示接水所需的总时间。

##### 样例输入

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

## 样例输出

4

### 5. 【基础】过河的最短时间

#### 题目描述

在漆黑的夜里，N位旅行者来到了一座狭窄而且没有护栏的桥边。如果不借助手电筒的话，大家是无论如何也不敢过桥去的。

不幸的是，N个人一共只带了一只手电筒，而桥窄得只够让两个人同时过。

如果各自单独过桥的话，N人所需要的时间已知；而如果两人同时过桥，所需要的时间就是走得比较慢的那个人单独行动时所需的时间。

问题是，如何设计一个方案，让这N人尽快过桥，计算成绩这N个人的最短过桥时间。

比如：有四个人甲乙丙丁，他们过河需要的时间分别为，甲：1 乙：2 丙：5 丁：10

第一种办法：最快的2个人先过桥，然后让跑的最快的人来回去接剩下的人：

先让甲乙过去（2分钟），甲回来（1分钟），甲丙过去（5分钟），甲回来（1分钟），甲丁再过去（10分钟），总共需要19分钟就可以让四个人都过去。

第二种办法：让最慢的2个人一起过桥，减少最慢的人在桥上的次数

先让甲乙过去（2分钟），甲回来（1分钟），丙丁过去（10分钟），乙回来（2分钟），甲乙再过去（2分钟），总共需要17分钟可以让四个人都过去。

那么最慢的时间就是需要17分钟！

#### 输入

每组测试数据的第一行是一个整数N( $1 \leq N \leq 1000$ )表示共有N个人要过河

每组测试数据的第二行是N个整数 $s_i$ ,表示这N个人过河所需要花时间。 $(0 < s_i \leq 100)$

#### 输出

所有人过河的最短时间

#### 样例输入

4

1 2 5 10

## 样例输出

17

#### 提示：

需要递推出如果只有一个人 ( $t_1$ ) 两个方案下的过河总时间，如果有2个人( $t_1, t_2$ )两个方案下的过河总时间，如果有3个人( $t_1, t_2, t_3$ )两个方案下的过河总时间...对比两个方案差异，找出规律

### 6. 【入门】购买贺年卡

#### 题目描述

新年快到了，笑笑打算给他的好朋友们发贺年卡，而且他已经选好了自己要购买的贺卡的样式。俗话说得好，货比三家，笑笑来到商店，看了各个商铺这种贺卡的价钱。不仅如此，笑笑还记住了每个商铺的存货量。已知笑笑打算购买m张贺卡，问他最少花多少钱。

#### 输入

第一行两个整数m和n。其中m表示要购买的贺卡的数量，n 表示商铺的个数。

以下n行，每行两个整数，分别表示该商铺这种贺卡的单价和存货量。

#### 输出

仅一个数，表示笑笑花的最少钱数。

#### 样例输入

```
10 4
4 3
6 2
8 10
3 6
```

#### 样例输出

```
36
```

## 第二十八章 递归算法

### 1. 递归算法的基础知识

#### 1.1 什么是递归？

递归：函数的自我调用

#### 1.2 递归的概念

数列递归：如果一个数列的项与项之间存在关联性，那么可以使用递归实现

原理：如果一个函数可以求 $A(n)$ ，那么该函数就可以求 $A(n-1)$ ，就形成了递归调用。

注意：一般起始项是不需要求解的，是已知条件。

#### 1.3 递归问题的求解过程

1. 找出规律
2. 函数调用自己求解前面的项
3. 交代起始项，让递归能够终止

#### 1.4 递归的重要思想

1. 既然函数能够求第 $n$ 项，那么它就能求第 $n-1$ 项，也能求第 $n+1$ 项（即要找到通用规律）
2. 既然函数能够解决一个问题的第 $n$ 步，就能解决第 $n-1$ 步，也能解决第 $n+1$ 步。

#### 1.5 递归算法解题的特点

1. 递归就是在过程或者在函数内调用自身。
2. 使用递归策略时，**必须要有一个明确的递归结束条件**，称作递归出口。
3. 递归算法解题通常显得很简洁，但是递归算法的解题运行效率却是很低的，所以一般不提倡使用递归。
4. 在递归调用的过程中系统为每一层的返回点、局部变量等开辟了栈来存储，递归的次数过多容易导致栈溢出等问题。

**小节：**由于递归的效率很低，且消耗内存，所以如果递归能够转化为循环，尽量使用循环。

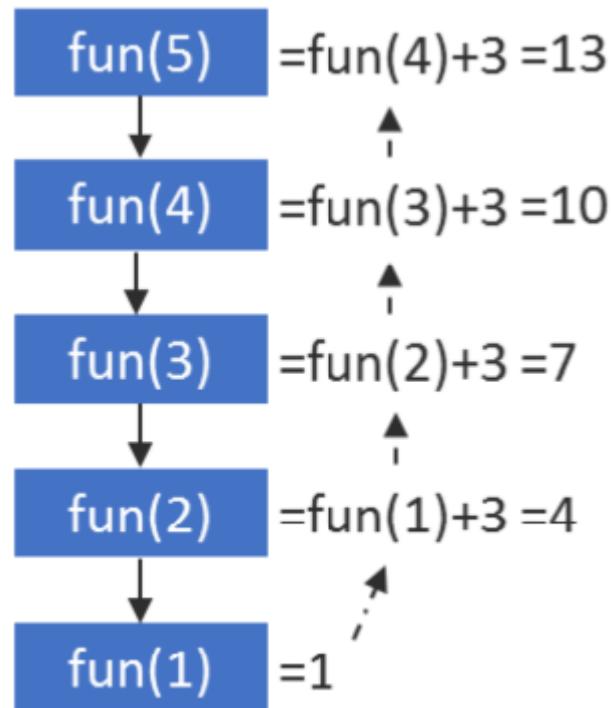
## 2. 入门案例

### 2.1 课堂案例

例：定义函数，递归求解等差数列1,4,7,10,13...的第n项的值。

规律： $A(n)=A(n-1)+3$

```
#include<bits/stdc++.h>
using namespace std;
int fun(int n){
    if(n==1){//递归出口
        return 1;
    }else{
        return fun(n-1)+3;
    }
}
int main(){
    int n;
    cin>>n;
    cout<<fun(n);
    return 0;
}
```



### 2.2 值传递与地址传递的区别

地址：变量在内存中的编号，如数组的本质是`a[0]`的地址

指针：c++中指针指的是地址

示例代码：

```

#include<bits/stdc++.h>
using namespace std;
void fun1(int x){
    x++;
}
void fun2(int a[]){
    a[0]++;
}
int main(){
    int x = 1;
    fun1(x);
    cout<<x<<endl;

    int a[3]={0};
    a[0]=1;
    cout<<a[0]<<endl;
    /*cout<<a<<endl;*/
    fun2(a);
    cout<<a[0]<<endl;

/*
//以0x开头表示16进制
int y=0x1A;
cout<<y<<endl;
//以0开头表示8进制
int z=017;
cout<<z<<endl;
*/
    return 0;
}

```

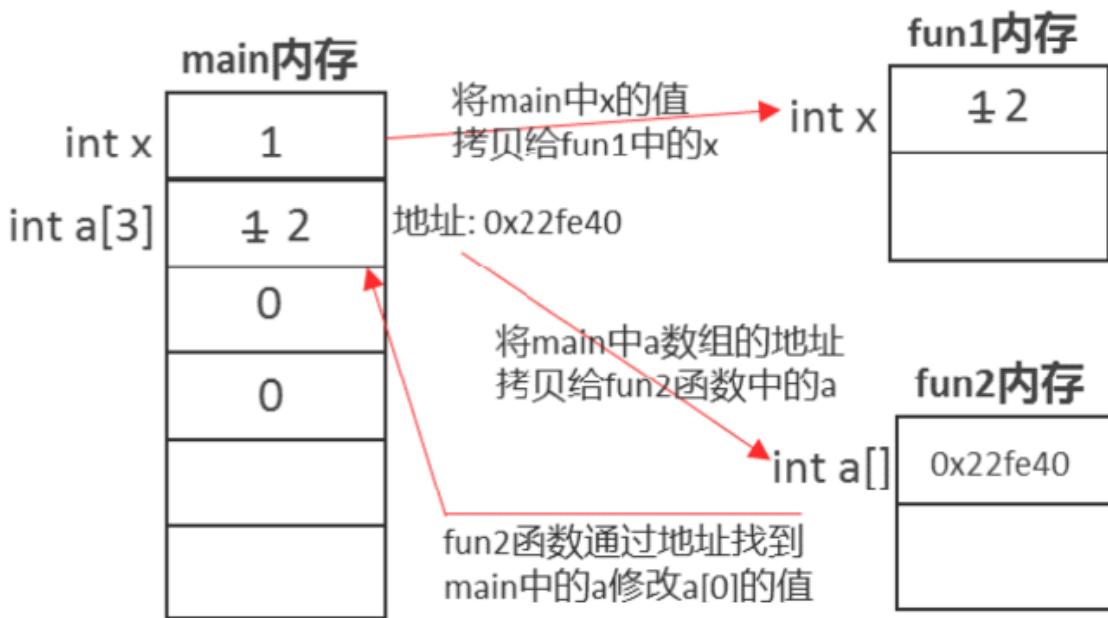
### 原理：

1. 每个函数运行时都会生成一个独立的内存来存储函数内部定义的变量，因此函数互相看不到对方内部定义的变量名，也不会出现命名冲突的情况；
2. 向函数中传递整数，本质是将整数的值拷贝给函数；向函数中传递数组，本质是将数组的地址拷贝给函数。

上述程序中，将main函数中的x拷贝给fun1中的x，实际上是拷贝的main函数中的x的**值**；将main函数中a拷贝给fun2中的a，实际拷贝的是main函数中a的**地址**。

**结论：**main函数中的x和fun1中的x不是同一个x，但main函数中的a和fun2中的a是同一个a（地址）。

如何判断的关键是看传递给函数的是一个整数值还是一个数组的地址。

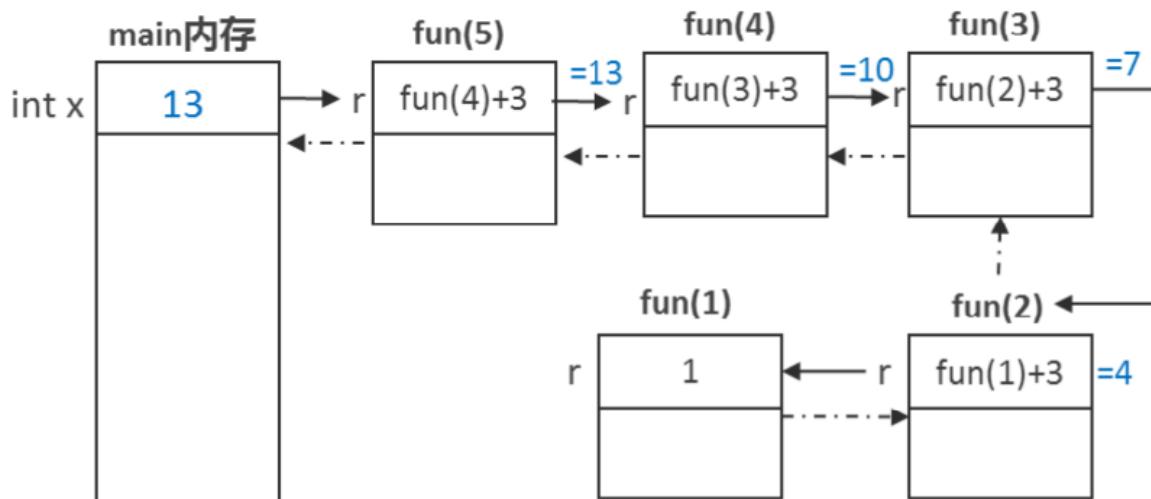


## 划定一块区域，供程序存储变量使用

等差数列递归过程中的内存储存情况：

示例代码

```
#include<bits/stdc++.h>
using namespace std;
//求首项为1, 差值为3的等差数列的第n项
int fun(int n){
    if(n==1){
        return 1;
    }else{
        return fun(n-1)+3;
    }
}
int main(){
    int x = fun(5);
    cout<<x;
    return 0;
}
```



### 3. 数值类递归问题练习

#### 3.1 课堂练习

##### 1. 【入门】编程求解 $1+2+3+\dots+n$

###### 题目描述

编程求解下列式子的值： $S=1+2+3+\dots+n$

###### 输入

输入一行，只有一个整数n( $1 \leq n \leq 1000$ )

###### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

###### 样例输入

```
100
```

###### 样例输出

```
5050
```

**解法一：**使用公共变量，将每次递归产生的变量加到总和上

###### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int n,s;
int fun(int i){
    if(i<=n){
        //cout<<i<<endl;
        s+=i;
        fun(i+1);
    }
}
int main(){
    cin>>n;
    fun(1);
    cout<<s;
    return 0;
}
```

###### 思想：

既然函数能够求第n项，那么它就能求第n-1项，也能求第n+1项。可以输出低n项，就可以输出低n+1项。

- 通过本问题可以让大家理解递归的本质（函数的自我调用），并从递归的功能角度先理解递归的作用。示例代码中fun()函数的作用是为了递归产生1~n中的每个数，换言之，既然函数能cout<<1，就能cout<<2 ... ...，这就是递归
- 能够听过fun()函数打印1~n的每一项，求和就很容易了

3. 能够理解局部变量和全局变量在递归中的区别：全局变量s和n在递归中值是可以共享的，而局部变量（如i）是函数独享的，需要注意的是，在递归中，如果是值不可以共享的变量，则不可以设置为公共的。

**解法二：**通过层层累加，层层返回求和

**示例代码：**

```
#include<bits/stdc++.h>
using namespace std;
int n;
int fun(int i){
    cout<<i<<endl;
    if(i<=n){
        return i+fun(i+1);
    }else{
        return 0;
    }
}
int main(){
    cin>>n;
    cout<<fun(1);
    return 0;
}
```

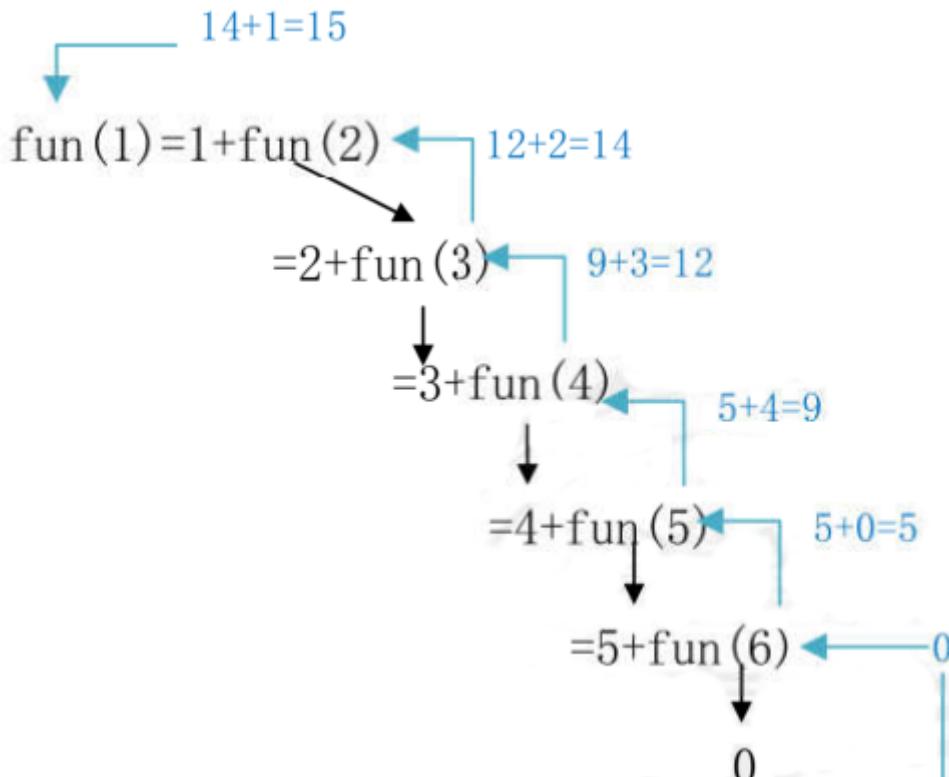
**重点思想：**

fun( $i \leq n$ )的和=1+fun( $i+1 \leq n$ )的和，因此fun()函数的目的是为了求1~n之间所有数的和。

1. 理解如何通过递归将值层层返回的过程

2. 本题的理解可以结合下图，如下图所示，求1~5的和，也就是fun(1)表示从1开始求和。

fun(1)=1+fun(2)=1+2+fun(3)=1+2+3+fun(4)=1+2+3+4+fun(5)=1+2+3+4+5+fun(6),由于 $6 > n$ ，所以递归终止，返回0，那么f(1)=1+2+3+4+5=15。



### 解法三：通过输入参数层层累加求和

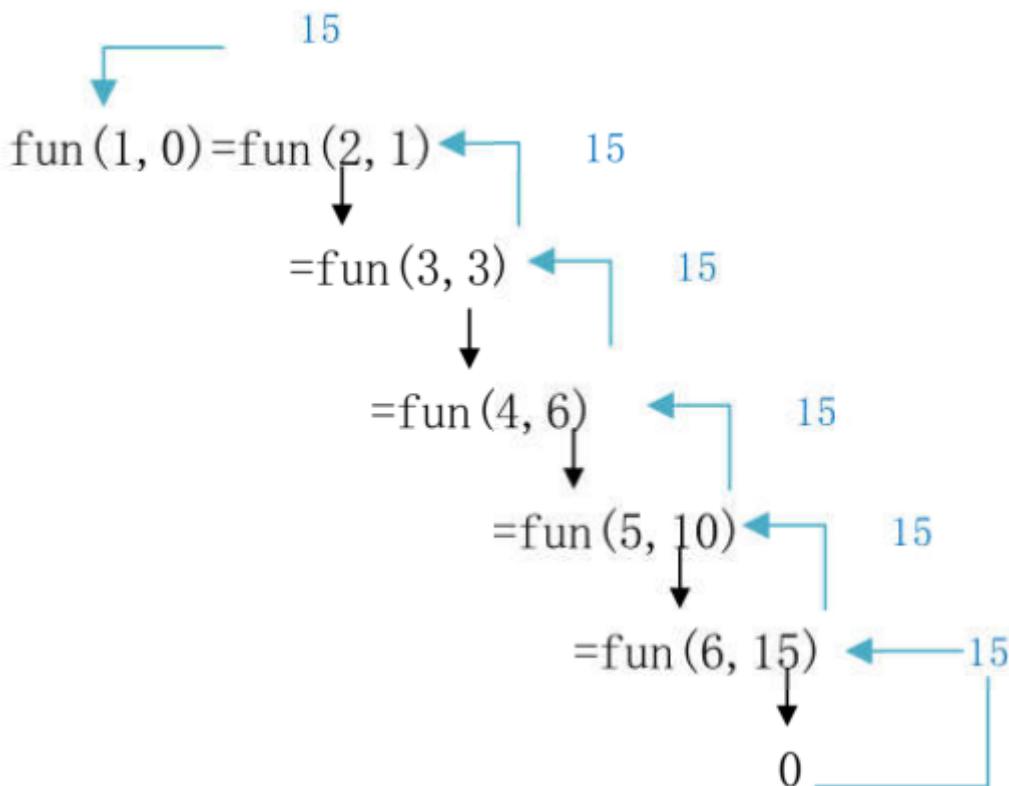
示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int n;
int fun(int i,int s){
    if(i<=n){
        return fun(i+1,s+i)
    }else{
        return s;
    }
}
int main(){
    cin>>n;
    cout<<fun(1,0);
    return 0;
}
```

重点思想：

通过fun()函数递归出1~n中的每个数，每遇到一个数就叠加到s上，然后层层递归深入，当递归结束，再层层返回求出s的值。

1. 理解通过输入参数累加求和的过程
2. 如下图所示，每一层在递归下一层时，都会把这一层得到的i加到总和s上去，并将s的值带到下一层，直到递归结束，s就是总和，此时层层返回s。
3. 和上一个解法不同，上一个解法每一层并未将得到的和带入下一层，而是在等待下一层的返回，在返回时计算总和，而在本解法中，是每一层都将和计算出来带入下一层继续计算，到递归结束时其实已经有结果了，只需要层层返回即可。



**说明：**本题采用三种解法并非累赘，而是在后续的深搜类问题中有相关的应用，所以务必掌握！

## 2. 【入门】角谷猜想

### 题目描述

日本一位中学生发现一个奇妙的定理，请角谷教授证明，而教授无能为力，于是产生了角谷猜想。猜想的内容：任给一个自然数，若为偶数则除以2，若为奇数则乘3加1，得到一个新的自然数后按上面的法则继续演算。若干次后得到的结果必为1。请编写代码验证该猜想：求经过多少次运算可得到自然数1。

如：输入22，则计算过程为。

22/2=11

11×3+1=34

34/2=17

17×3+1=52

52/2=26

26/2=13

13×3+1=40

40/2=20

20/2=10

10/2=5

5×3+1=16

16/2=8

8/2=4

4/2=2

2/2=1

经过15次运算得到自然数1。

### 输入

一行，一个正整数n。 (1 <= n <= 20000 )

### 输出

一行，一个整数，表示得到1所用的运算次数。

### 样例输入

```
22
```

### 样例输出

```
15
```

**解法一：**通过公共变量累计总次数

**示例代码：**

```
#include<bits/stdc++.h>
using namespace std;
int c;
void fun(int n){
    if(n!=1){
        if(n%2==0){
            fun(n/2);
        }else{
```

```

        fun(n*3+1);
    }
    c++;
}
}

int main(){
    int n;
    cin>>n;
    fun(n);
    cout<<c;
    return 0;
}

```

**解法二：** 累计计算递归次数，层层返回

**示例代码：**

```

#include<bits/stdc++.h>
using namespace std;
int fun(int n){
    if(n!=1){
        if(n%2==0){
            return 1+fun(n/2);
        }else{
            return 1+fun(3*n+1);
        }
    }else{
        return 0;
    }
}
int main(){
    int n;
    cin>>n;
    cout<<fun(n);
    return 0;
}

```

**解法三：** 通过输入参数逐层+1.到最后返回

**示例代码：**

```

#include<bits/stdc++.h>
using namespace std;
int fun(int n,int c){
    if(n!=1){
        if(n%2==0){
            return fun(n/2,1+c);
        }else{
            return fun(3*n+1,1+c);
        }
    }else{
        return c;
    }
}
int main(){

```

```
int n;
cin>>n;
cout<<fun(n,0);
return 0;
}
```

### 3. 【入门】正整数N转换成一个二进制数

#### 题目描述

输入一个不大于32767的整数n，将它转换成一个二进制数。

#### 输入

输入只有一行，包括一个整数n(0<=n<=32767)

#### 输出

输出只有一行。

#### 样例输入：

```
100
```

#### 样例输出：

```
1100100
```

#### 解法一：利用公共变量累加

#### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
string s;
void fun(int n){
    char c;
    if(n!=0){
        c=n%2+'0';
        s=c+s;
        fun(n/2);
    }
}
int main(){
    int n;
    cin>>n;
    fun(n);
    if(n==0){
        cout<<0;
    }else{
        cout<<s;
    }
    return 0;
}
```

#### 解法二：累计计算递归次数，层层返回

#### 示例代码：

```

#include<bits/stdc++.h>
using namespace std;
string fun(int n){
    char c;
    if(n!=0){
        c=n%2+'0';
        return fun(n/2)+c;
    }else{
        return "";
    }
}
int main(){
    int n;
    cin>>n;
    if(n==0){
        cout<<0;
    }else{
        cout<<fun(n);
    }
    return 0;
}

```

**解法三：**通过输入参数逐层+1.到最后返回

**示例代码：**

```

#include<bits/stdc++.h>
using namespace std;
string fun(int n,string r){
    char c;
    if(n!=0){
        c=n%2+'0';
        return fun(n/2,c+r);
    }else{
        return r;
    }
}
int main(){
    int n;
    cin>>n;
    if(n==0){
        cout<<0;
    }else{
        cout<<fun(n,"");
    }
    return 0;
}

```

#### 4. 【入门】求两个自然数M和N的最大公约数

**题目描述**

求两个自然数M和N的最大公约数(M, N都在长整型范围内)

**输入**

输入一行，包括两个正整数

**输出**

输出只有一行，包括1个正整数。

### 样例输入

```
45 60
```

### 样例输出

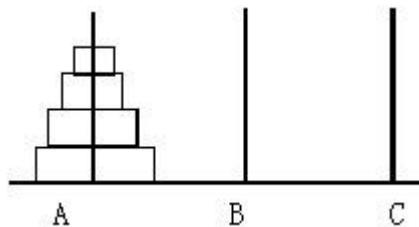
```
15
```

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
long long fun(long long a, long long b){
    //辗转相除法，重复递归求解
    if(a%b!=0){
        return fun(b, a%b);
    }else{
        return b;
    }
}
int main(){
    /*
    int a,b,t;
    cin>>a>>b;
    while(a%b!=0){
        t=a%b;
        a=b;
        b=t;
    }
    cout<<b;
    */
    long long a,b;
    cin>>a>>b;
    cout<<fun(a,b);
    return 0;
}
```

## 5. 【入门】汉诺塔的移动次数

### 题目描述



汉诺塔（又称河内塔）问题是印度的一个古老的传说。开天辟地的神勃拉玛在一个庙里留下了三根金刚石的棒，第一根上面套着64个圆的金片，最大的一个在底下，其余一个比一个小，依次叠上去，庙里的众僧不倦地把它们一个个地从这根棒搬到另一根棒上，规定可利用中间的一根棒作为帮助，但每次只能搬一个，而且大的不能放在小的上面。面对庞大的数字(移动圆片的次数)18446744073709551615，看来，众僧们耗尽毕生精力也不可能完成金片的移动。

后来，这个传说就演变为汉诺塔游戏：

- 1.有三根杆子A,B,C。A杆上有若干碟子
- 2.每次移动一块碟子,小的只能叠在大的上面
- 3.把所有碟子从A杆全部移到C杆上

请问n个金片的情况下，需要最少移动多少次？

#### 输入

输入一个整数n代表金片的数量 (n<=20)

#### 输出

一个整数，代表n个金片的移动次数

#### 样例输入

```
3
```

#### 样例输出

```
7
```

**思路：**先从一两个金片的简单情况开始推导找出规律: $f(n)=f(n-1)*2+1$

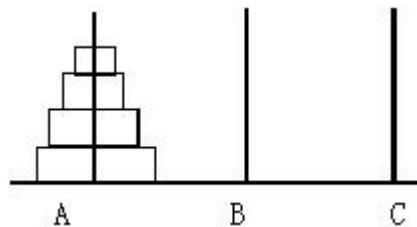
**示例代码：**

```
#include<bits/stdc++.h>
using namespace std;
//递归求解
int fun(int n){
    //递归的出口
    if(n==1){
        return 1;
    } else{
        return fun(n-1)*2+1;
    }
}
int main(){
/*
1. 先将n-1个金片(n个金片除了最下面一片之外所有的)
从A的位置借助C的位置，移动到B的位置，需要fun(n-1)步，
2. 将A位置的最下面的一个金片直接移动到 C的位置，需要1步
3. 再将B位置的n-1个金片，从B位置借助A位置移动到 C位置，
需要fun(n-1)步
因此结论如下： fun(n)=fun(n-1)*2+1;
*/
    int n;
    cin>>n;
    cout<<fun(n);
    return 0;
}
```

## 6. 【基础】经典递归问题——汉诺塔

#### 题目描述

汉诺塔（又称河内塔）问题是印度的一个古老的传说。开天辟地的神勃拉玛在一个庙里留下了三根金刚石的棒，第一根上面套着64个圆的金片，最大的一个在底下，其余一个比一个小，依次叠上去，庙里的众僧不倦地把它们一个个地从这根棒搬到另一根棒上，规定可利用中间的一根棒作为帮助，但每次只能搬一个，而且大的不能放在小的上面。面对庞大的数字(移动圆片的次数)18446744073709551615，看来，众僧们耗尽毕生精力也不可能完成金片的移动。



后来，这个传说就演变为汉诺塔游戏：

- 1.有三根杆子A,B,C。A杆上有若干碟子
- 2.每次移动一块碟子,小的只能叠在大的上面
- 3.把所有碟子从A杆全部移到C杆上

经过研究发现，汉诺塔的破解很简单，就是按照移动规则向一个方向移动金片：

如3阶汉诺塔的移动：A→C,A→B,C→B,A→C,B→A,B→C,A→C      此外，汉诺塔问题也是程序设计中的经典递归问题。

算法思路： 1.如果只有一个金片，则把该金片从源移动到目标棒，结束。  
2.如果有n个金片，则把前n-1个金片移动到辅助的棒，然后把自己移动到目标棒，最后再把前n-1个移动到目标棒。

#### 输入

一个整数N，表示A柱上有N个碟子。 (0<n<=10)

#### 输出

若干行，即移动的最少步骤

#### 样例输入

```
3
```

#### 样例输出

```
A To C
A To B
C To B
A To C
B To A
B To C
A To C
```

#### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
/*
    函数的作用是将n个金片从p1的位置借助p2的位置移动到p3位置
    p1:金片原位置
    p2:辅助位置
    p3:目标位置
*/
```

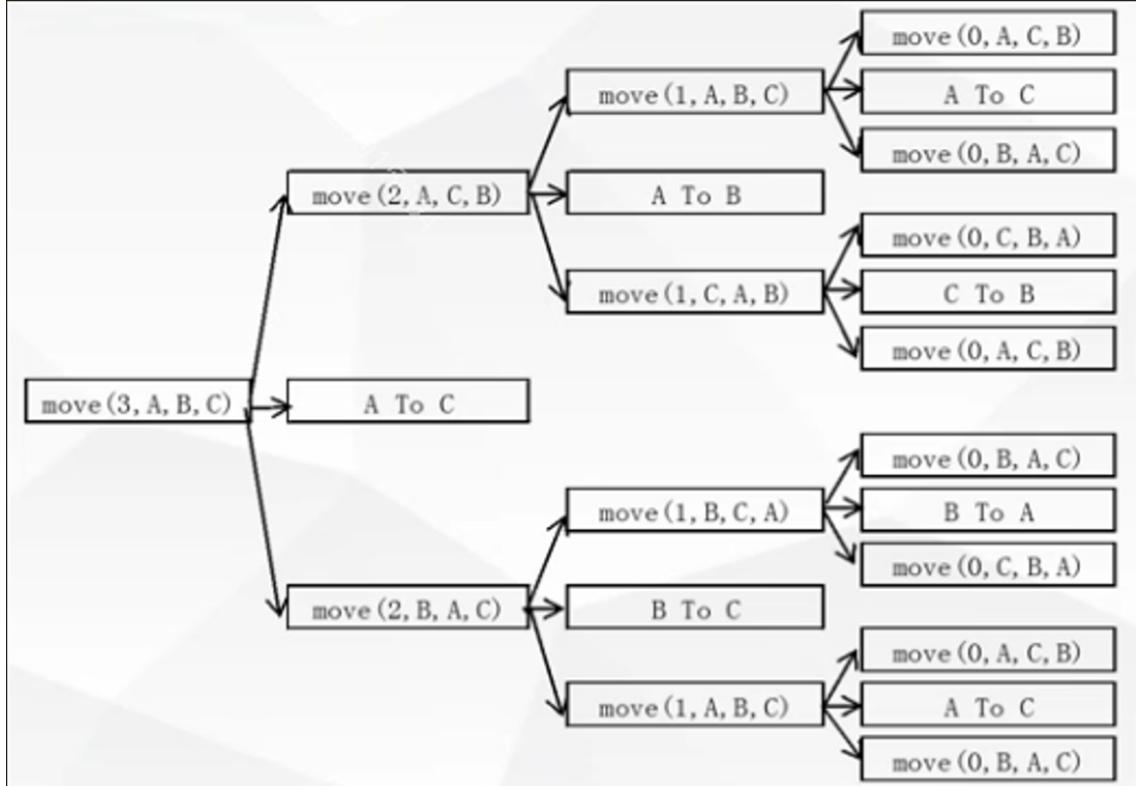
```

void move(int n, char p1, char p2, char p3){
    //递归的出口，只要有金片就递归
    if(n>0){
        //第一步，将n-1个金片从p1位置通过p3位置移动到p2位置
        move(n-1, p1, p3, p2);
        //第二步，将p1位置的第n个金片直接移动到p3的位置
        cout<<p1<<" TO "<<p3<<endl;
        //第三步，将p2位置的n-1个金片借助p1移动到p3位置
        move(n-1, p2, p1, p3);
    }
}

int main(){
    /*
    1.先将n-1个金片(n个金片除了最下面一片之外所有的)
    从A的位置借助C的位置，移动到B的位置，需要fun(n-1)步,
    2. 将A位置的最下面的一个金片直接移动到 C的位置，需要1步
    3.再将B位置的n-1个金片，从B位置借助A位置移动到 C位置,
    需要fun(n-1)步
    因此结论如下: fun(n)=fun(n-1)*2+1;
    */
    int n;
    cin>>n;
    move(n, 'A', 'B', 'C');
    return 0;
}

```

递归过程：



## 3.2 课后作业

### 1. 【基础】回文数

#### 题目描述

回文数的定义为：如果把一个数的各个数位上的数字颠倒过来得到的新数与原数相等，则此数是回文数，例：7,22,131,2112,31013,...都是回文数。对任意给出的一个整数n,经过一系列的处理，最后都能成为回文数。处理的方法是，该数加上它的颠倒数，

例如：n=176

第一次处理后       $176+671 = 847$

第二次处理后       $847+748 = 1595$

第三次处理后       $1595+5951 = 7546$

第四次处理后       $7546+6457 = 14003$

第五次处理后       $14003+30041 = 44044$

此时成为回文数，共进行 5 次处理。

问题：给出n 后，求出使该数按照以上规则进行一系列处理后成为回文数的最少操作次数。

#### 输入

n 一个整数 ( $1 \leq n \leq 1000000$ )

#### 输出

使n成为回文数的最少处理次数。若开始给出的n是回文数，则输出 0 (即不需任何处理)。

#### 样例输入

67

#### 样例输出

2

### 2. 【入门】因子求和

#### 题目描述

已知一个正整数N ( $20 \leq N \leq 800000000$ )，请你编写程序求出该数的全部因子（不包括1和n）的和

#### 输入

一个正整数n。

#### 输出

一个整数代表n的因子和。

#### 样例输入

24

#### 样例输出

35

### 3. 【入门】请问一个正整数能够整除几次2?

#### 题目描述

请问一个正整数n能够整除几次2?

比如: 4可以整除2次2, 100可以整除2次2, 9可以整除0次2。

#### 输入

从键盘读入一个正整数n

#### 输出

输出一个整数, 代表n能够整除2的次数

#### 样例输入

8

#### 样例输出

3

### 4. 【基础】数的计数

#### 题目描述

输入一个自然数n ( $n \leq 1000$ ) , 然后对自然数按照如下方法进行处理:

在该自然数的左侧加上一个自然数, 但加上的数不能超过n的一半;

加上数后继续按此规则处理, 直到不能再添加自然数为止;

请问按照这样的方法添加数, 能够产生多少个新数?

例如: n=6, 则左侧添加数的方案有

16

26

126

36

136

共能够产生5个新数。

#### 输入

一个整数n ( $n \leq 100$ )

#### 输出

按照规则能够产生的新数的个数

#### 样例输入

6

#### 样例输出

5

### 5. 【入门】两个自然数M和N的最小公倍数

#### 题目描述

求两个整数M和N的最小公倍数。

#### 输入

输入一行, 包括两个整数.

#### 输出

输出只有一行（这意味着末尾有一个回车符号），包括1个整数。

#### 样例输入

```
45 60
```

#### 样例输出

```
180
```

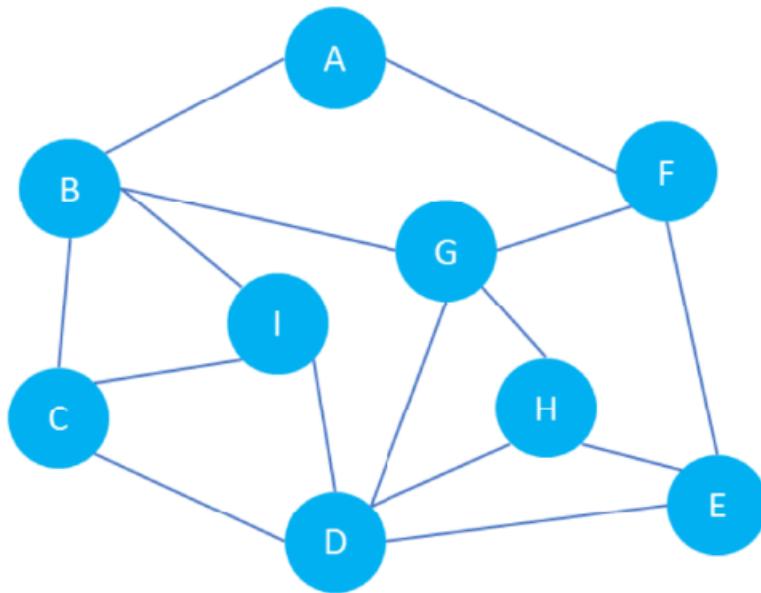
## 第二十九章 深搜（DFS）与回溯

### 1. 深搜的基础知识

#### 1.1 什么是深搜？

深度优先搜索属于图算法的一种，DFS即Depth First Search。过程是对每一个可能的分支路径深入到不能再深入为止，而每一个节点只能访问一次。

案例：图的遍历



遍历图中部分节点之间所有有路径连接的点，规则为：深度优先，右手路径优先，不能重复遍历，请问遍历的结果是什么？

遍历结果:ABCDEF

### 1.2 课堂练习

#### 1. 【入门】扫地机器人

##### 题目描述

小兰同学在为扫地机器人设计一个在矩形区域中行走的算法，小兰是这样设计的：先把机器人放在出发点(1,1)点上，机器人在每个点上都会沿用如下的规则来判断下一个该去的点是哪里。规则：优先向右，如果向右不能走（比如：右侧出了矩形或者右侧扫过了）则尝试向下，向下不能走则尝试向左，向左不能走则尝试向上；直到所有的点都扫过。

小兰为了验证自己设计的算法是否正确，打算先模拟一下这个算法，每当机器人走过一个单元格时，会在单元格内标记一个数字，这个数字从1开始，每经过一个单元格数字会递增1，直到所有的单元格都扫一遍，也就是所有的单元格都标记过数字，机器人会自动停止。

比如：如果机器人按照上面的规则，清扫一个 $3 * 4$ 大小的矩形区域，那么标记数字的结果如下图所示。

	1	2	3	4
1	1	2	3	4
2	10	11	12	5
3	9	8	7	6

再比如：如果机器人按照上面的规则，清扫一个 $5 * 6$ 大小的矩形区域，那么标记数字的结果如下图所示。

	1	2	3	4	5
1	1	2	3	4	5
2	22	23	24	25	6
3	21	20	19	18	7
4	14	15	16	17	8
5	13	12	11	10	9

请你帮助小兰设计一个程序，按照上面的规则，将一个 $n * m$ 大小的矩形，标记一下数字，输出最终标记的结果。

### 输入

一行内有2个两个整数n和m，用空格隔开，分别代表矩形区域的行数（高）和列数（宽）（n和m都是2~9之间的整数）

### 输出

输出按题意机器人走过每个点之后，标记数字的结果，每个数字输出时场宽设置为3。

### 样例输入

```
3 4
```

### 样例输出

```
1 2 3 4
10 11 12 5
9 8 7 6
```

**思路：**本题求为每个点赋值，可以定义一个函数来为矩阵赋值，比如定义函数fun(int x,int y,int k)，作用为将x、y点赋值为k。

调用函数时，首先为 $1,1$ 点赋值为1，即fun(1,1,1)，既然函数能为 $1,1$ 赋值，就能为 $1,2$ 赋值，以此类推，为典型的递归问题。需要注意的是，每次赋值结束，所赋的值都要+1.

本题需要掌握以下内容：

1. 理解赋值的顺序，任意给定一个n\*m的矩阵，都可以写出赋值的结果
2. 理解递归的过程，任意给定一个n\*m的矩阵，都可以画出递归的过程
3. 理解程序的编写过程及实现原理
4. 理解函数递归调用的过程
5. 理解迷宫类递归的几种不同写法

**解法一：**从出发点开始，探测每个点，标记数字，标记结束后，分别递归四个方向，如果探测点满足要求，则访问该点。

```
#include<bits/stdc++.h>
using namespace std;
int n,m,a[20][20];
void fun(int x,int y,int k){
    a[x][y]=k;
    //按照优先级判断是否可以访问
    if(y+1<=m&&a[x][y+1]==0) fun(x,y+1,k+1);
    if(x+1<=n&&a[x+1][y]==0) fun(x+1,y,k+1);
    if(y-1>=1&&a[x][y-1]==0) fun(x,y-1,k+1);
    if(x-1>=1&&a[x-1][y]==0) fun(x-1,y,k+1);
}
int main(){
    int i,j;
    //设定矩阵
    cin>>n>>m;
    //开启访问
    fun(1,1,1);
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}
```

**解法二：**从出发点开始，探索每个点，标记数字，标记结束后，分别递归四个方向，不管探测点是否可访问，直接递归，在递归进入函数的时候判断递归点是否可访问。

```
#include<bits/stdc++.h>
using namespace std;
int n,m,a[20][20];
void fun(int x,int y,int k){
    //由于是直接访问赋值，所以可能有的点不可访问
    //在这里进行判断
    if(x<=n&&y<=m&&x>=1&&y>=1&&a[x][y]==0){
        a[x][y]=k;
        //    向右
        fun(x,y+1,k+1);
        //    向下
        fun(x+1,y,k+1);
        //    向左
        fun(x,y-1,k+1);
    }
}
```

```

//      向上
    fun(x-1,y,k+1);
}
}

int main(){
    int i,j;
    //设定矩阵
    cin>>n>>m;
    //开启访问
    fun(1,1,1);
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}

```

**解法三：**使用方向数组（即通过对解法二发现上下左右仅仅是x、y加减了不同的值而已，所以可以将x、y的变化定义为两个数组），循环递归访问四个方向。（也可以氛围在探测前判断即将探测点的可访问情况以及无论是否可访问直接递归，在进入函数时判断是否可访问两种方式）

```

#include<bits/stdc++.h>
using namespace std;
int n,m,a[20][20];
int fx[5]={0,0,1,0,-1};
int fy[5]={0,1,0,-1,0} ;
void fun(int x,int y,int k){
    //由于是直接访问赋值，所以可能有的点不可访问
    //在这里进行判断
    if(x<=n&&y<=m&&x>=1&&y>=1&&a[x][y]==0){
        a[x][y]=k;
        int tx,ty;//表示要去的点
        for(int i=1;i<5;i++){
            tx=x+fx[i];
            ty=y+fy[i];
            fun(tx,ty,k+1);
        }
    }
}
int main(){
    int i,j;
    //设定矩阵
    cin>>n>>m;
    //开启访问
    fun(1,1,1);
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            cout<<setw(3)<<a[i][j];
        }
        cout<<endl;
    }
    return 0;
}

```

```
}
```

### 注意：

上述方法可以在地柜之前判断tx和ty是否为有效坐标，修改方法如下：

```
void fun(int x,int y,int k){  
    a[x][y]=k;  
    int tx,ty;//表示要去的点  
    for(int i=1;i<5;i++){  
        tx=x+fx[i];  
        ty=y+fy[i];  
        if(tx<=n&&ty<=m&&tx>=1&&ty>=1&&a[tx][ty]==0){  
            fun(tx,ty,k+1);  
        }  
    }  
}
```

## 2. 【基础】迷宫出口

### 题目描述

一天小宏在森林里探险的时候不小心走入了一个迷宫，迷宫可以看成是由 $n \times n$ 的格点组成，每个格点只有2种状态，0和1，前者表示可以通行后者表示不能通行。同时当小宏处在某个格点时，他只能移动到东南西北(或者说上下左右)四个方向之一的相邻格点上，小宏想要从点A走到点B，问在不走出迷宫的情况下能不能办到。如果起点或者终点有一个不能通行(为1)，则看成无法办到。

### 输入

第1行是一个正整数 $n$  ( $1 \leq n \leq 100$ )，表示迷宫的规模是 $n \times n$ 的。接下来是一个 $n \times n$ 的矩阵，矩阵中的元素为0或者1。再接下来一行是4个整数 $ha\ la\ hb\ lb$ ，描述A处在第 $ha$ 行 第 $la$ 列，B处在第 $hb$ 行 第 $lb$ 列。

### 输出

能办到则输出“YES”，否则输出“NO”。

### 样例输入

```
3  
0 1 1  
0 0 1  
1 0 0  
1 1 3 3
```

### 样例输出

```
YES
```

**思路：**从出发点开始，探测所有可探测的点，看是否包含目标点，如果有就表示可达，没有就表示不可达。

	j=1	j=2	j=3
i=1	0	0	0
i=2	0	1	1
i=3	0	0	0

探索的顺序：右、下、左、上

假设从 1, 1 点，探索到 3, 3 点

解决办法一：

```
#include<bits/stdc++.h>
using namespace std;
int n,a[101][101];
int s1,s2,e1,e2;
int fx[5]={0,0,1,0,-1};
int fy[5]={0,1,0,-1,0};
bool f=false;
void fun(int x,int y){
    a[x][y]=1;//防止死循环
    int tx,ty;//表示要去的点
    for(int i=1;i<5;i++){
        tx=x+fx[i];
        ty=y+fy[i];
        if(tx<=n&&ty<=n&&tx>=1&&ty>=1&&a[tx][ty]==0){
            if(tx==e1&&ty==e2){
                f=true;
            }else{
                fun(tx,ty);
            }
        }
    }
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            cin>>a[i][j];
        }
    }
    cin>>s1>>s2>>e1>>e2;
    if(a[s1][s2]==1||a[e1][e2]==1){
        cout<<"NO";
    }else{
        fun(s1,s2);
        if(f){
            cout<<"YES";
        }else{
            cout<<"NO";
        }
    }
}
```

```
    }
}
return 0;
}
```

问：上述解法在到达目标之后是否停止了？

解决办法一：在递归条件中加入f==false，要求没有找到终点才递归，进而减少不必要的递归

```
#include<bits/stdc++.h>
using namespace std;
int n,a[101][101];
int s1,s2,e1,e2;
int fx[5]={0,0,1,0,-1};
int fy[5]={0,1,0,-1,0};
bool f=false;
void fun(int x,int y){
    a[x][y]=1;//防止死循环
    int tx,ty;//表示要去的点
    for(int i=1;i<5;i++){
        tx=x+fx[i];
        ty=y+fy[i];
        if(tx<=n&&ty<=n&&tx>=1&&ty>=1&&a[tx][ty]==0&&!f){
            if(tx==e1&&ty==e2){
                f=true;
            }else{
                fun(tx,ty);
            }
        }
    }
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            cin>>a[i][j];
        }
    }
    cin>>s1>>s2>>e1>>e2;
    if(a[s1][s2]==1||a[e1][e2]==1){
        cout<<"NO";
    }else{
        fun(s1,s2);
        if(f){
            cout<<"YES";
        }else{
            cout<<"NO";
        }
    }
    return 0;
}
```

## 解决办法二：在找到终点时，通过exit(0)来终止程序

```
#include<bits/stdc++.h>
using namespace std;
int n,a[101][101];
int s1,s2,e1,e2;
int fx[5]={0,0,1,0,-1};
int fy[5]={0,1,0,-1,0};
void fun(int x,int y){
    a[x][y]=1;//防止死循环
    int tx,ty;//表示要去的点
    for(int i=1;i<5;i++){
        tx=x+fx[i];
        ty=y+fy[i];
        if(tx<=n&&ty<=n&&tx>=1&&ty>=1&&a[tx][ty]==0){
            if(tx==e1&&ty==e2){
                cout<<"YES";
                exit(0);
            }else{
                fun(tx,ty);
            }
        }
    }
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            cin>>a[i][j];
        }
    }
    cin>>s1>>s2>>e1>>e2;
    if(a[s1][s2]==1||a[e1][e2]==1){
        cout<<"NO";
    }else{
        fun(s1,s2);
        cout<<"NO";
    }
    return 0;
}
```

### 3. 【基础】数池塘（四方向）

#### 题目描述

农夫博丹的农场可以表示成N\*M ( $1 \leq N \leq 100 \leq M \leq 100$ ) 个方格组成的矩形。由于近日的降雨，在博丹农场上的不同地方形成了池塘。每一个方格或者有积水 ('W') 或者没有积水 ('.')。农夫博丹打算数出他的农场上共形成了多少池塘。一个池塘是一系列相连的有积水的方格，每一个方格周围的四个方格都被认为是与这个方格相连的。现给出博丹农场的图样，要求输出农场上的池塘数。

#### 输入

第1行：由空格隔开的两个整数：N和M

第2..N+1行：每行M个字符代表博丹农场的一排方格的状态。每个字符或者是'W'或者是'.'，字符之间没有空格。

### 输出

输出只有1行，输出博丹农场上的池塘数

### 样例输入：

```
10 12
W.....WW.
.WWW.....WWW
...WW...WW...
.....WW...
.....W...
..W....W...
.W.W.....WW.
W.W.W.....W.
.W.W.....W.
..W.....W.
```

### 样例输出

```
13
```

### 思路：

	j=1	j=2	j=3	j=4
i=1	0	0	0	0
i=2	0	0	0	0
i=3	0	0	0	0
i=4	0	0	0	0

循环每个点，如果当前的点是"w"，计数器加1，然后从当前的i,j开始递归，将上下左右中相邻的为w的都标记为"."

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int m,n;
char a[101][101];
int fx[5]={0,0,1,0,-1};
int fy[5]={0,1,0,-1,0};
void dfs(int x,int y){
    a[x][y]='.';
    int tx,ty;
    for(int i=1;i<5;i++){
        tx=x+fx[i];
        ty=y+fy[i];
```

```

        if(a[tx][ty]=='W'){
            dfs(tx,ty);
        }
    }

int main(){
    cin>>n>>m;
    int i,j,c=0;
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            cin>>a[i][j];
        }
    }
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            if(a[i][j]=='W'){
                dfs(i,j);
                c++;
            }
        }
    }
    cout<<c;
    return 0;
}

```

## 1.3 课后作业

### 1. 【基础】数池塘（八方向）

#### 题目描述

农夫博丹的农场可以表示成 $N \times M$  ( $1 \leq N \leq 100 \leq M \leq 100$ ) 个方格组成的矩形。由于近日的降雨，在博丹农场上的不同地方形成了池塘。每一个方格或者有积水 ('W') 或者没有积水 ('.')。农夫博丹打算数出他的农场上共形成了多少池塘。一个池塘是一系列相连的有积水的方格，每一个方格周围的八个方格都被认为是与这个方格相连的。现给出博丹农场的图样，要求输出农场上的池塘数。

#### 输入

第1行：由空格隔开的两个整数：N和M

第2..N+1行：每行M个字符代表博丹农场的一排方格的状态。每个字符或者是'W'或者是'.'，字符之间没有空格。

#### 输出

输出只有1行，输出博丹农场上的池塘数

#### 样例输入

10 12

```
W.....WW.  
.WWW.....WWW  
....WW...WW.  
.....WW.  
.....W..  
.W.....W..  
.WW.....WW.  
WW.W.....W.  
.WW.....W.  
..W.....W.
```

### 样例输出

3

## 2. 【提高】奶牛和草丛

### 题目描述

奶牛Bessie计划好好享受柔软的春季新草。新草分布在R行C列的牧场里。它想计算一下牧场中的草丛数量。

在牧场地图中，每个草丛要么是单个“#”，要么是有公共边的相邻多个“#”。给定牧场地图，计算有多少个草丛。

例如，考虑如下5行6列的牧场地图

```
.#....  
..#...  
..#..#  
...##  
....#
```

这个牧场有3个草丛：一个在第一行，一个在第二列横跨了二、三行，一个在第三行横跨了三、四、五行。

### 输入

第一行包含两个整数R和C，中间用单个空格隔开。

接下来R行，每行C个字符，描述牧场地图。字符只有“#”或“.”两种。 $(1 \leq R, C \leq 100)$

### 输出

输出一个整数，表示草丛数。

### 样例输入

5 6

```
.#....  
..#...  
..#..#  
...##  
....#
```

### 样例输出

3

## 2. 深搜的进阶问题

### 2.1 最少步数问题

#### 【基础】走出迷宫的最少步数

##### 题目描述

一个迷宫由R行C列格子组成，有的格子里有障碍物，不能走；有的格子是空地，可以走。

给定一个迷宫，求从左上角走到右下角最少需要走多少步(数据保证一定能走到)。只能在水平方向或垂直方向走，不能斜着走。

##### 输入

第一行是两个整数，R和C，代表迷宫的行数和列数。（ $1 \leq R, C \leq 40$ ）

接下来是R行，每行C个字符，代表整个迷宫。空地格子用'.'表示，有障碍物的格子用'#'表示。迷宫左上角和右下角都是'!'。

##### 样例输入

```
5 5
..####
#.....
#.##.#
#.##.#
#.##..
```

##### 样例输出

```
9
```

	j=1	j=2	j=3	j=4
i=1	*	*	*	*
i=2	*	*	*	*
i=3	*	#	#	#
i=4	*	*	*	*

s数组：存迷宫

	j=1	j=2	j=3	j=4
i=1	1	2	3	4
i=2	2	3	4	5
i=3	3	#	#	#
i=4	4	5	6	7

d数组：每个点最少步数

**思路：**如果走到某个点x、y，需要的步数，比d数组中记录的最少的步数还少，则走该点。d数组的初始值设置为INT\_MAX，递归函数设计：dfd(x,y,k)，在递归时，dfs(tx,ty,k+1)

1. 准备一个整数数组，记录从出发点到每个点至少需要多少步，初始化为INT\_MAX
2. 从出发点开始探索，顺时针探索，若该点可达，且到该点的步数更少，则替代d数组的步数
3. 最终d数组记录了到每个点至少需要多少步，a[n][m]就是最终结果

**注意：**递归的重点是如何防止死循环。

**示例代码：**

```

#include<bits/stdc++.h>
using namespace std;
int m,n;
char a[50][50];//地图
int d[50][50];//存储走到每个店最少需要多少步
//方向值变化的数组
int fx[5] = {0,0,1,0,-1};
int fy[5] = {0,1,0,-1,0};
//递归探索地图，求走到每个点最少需要多少步
void dfs(int x,int y,int dep){
    d[x][y]=dep;
    int tx,ty;
    for(int i=1;i<5;i++){
        tx=x+fx[i];
        ty=y+fy[i];
        if(a[tx][ty]=='.' && dep+1<d[tx][ty]){
            dfs(tx,ty,dep+1);
        }
    }
}
int main(){
    int i,j;
    cin>>n>>m;
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            cin>>a[i][j];
            d[i][j]=INT_MAX;
        }
    }
    dfs(1,1,1);
    cout<<d[n][m];
    return 0;
}

```

## 2.2 第一条路径问题

### 【基础】迷宫的第一条出路

#### 题目描述

已知一 $N \times N$ 的迷宫，允许往上、下、左、右四个方向行走，现请你按照左、上、右、下顺序进行搜索，找出第一条从左上角到右下角的路径。

#### 输入

输入数据有若干行，第一行有一个自然数 $N$  ( $N \leq 20$ )，表示迷宫的大小，其后有 $N$ 行数据，每行有 $N$ 个0或1（数字之间没有空格，0表示可以通过，1表示不能通过），用以描述迷宫地图。入口在左上角 (1, 1) 处，出口在右下角 ( $N$ ,  $N$ ) 处。所有迷宫保证存在从入口到出口的可行路径。

#### 输出

输出数据仅一行，为按照要求的搜索顺序找到的从入口到出口的第一条路径（搜索顺序：左、上、右、下）。

#### 样例输入

```
4  
0001  
0100  
0010  
0110
```

### 样例输出

```
(1,1) -> (1,2) -> (1,3) -> (2,3) -> (2,4) -> (3,4) -> (4,4)
```

思路：

解法一：

示例代码：

```
#include<bits/stdc++.h>  
using namespace std;  
//左上右下顺序  
char a[30][30];  
int r[410][3];//记录正确的第一条路径  
int n;  
//方向的变化  
int fx[5]={0,0,-1,0,1};  
int fy[5]={0,-1,0,1,0};  
void print(int k){  
    for(int i=1;i<=k;i++){  
        cout<<"("<<r[i][1]<<","<<r[i][2]<<")";  
        //如果不是最后一个点，打印连接的-->  
        if(i!=k){  
            cout<<"-->";  
        }  
    }  
    exit(0); //停止程序  
}  
int dfs(int x,int y,int k){  
    //记录探索到的点的坐标  
    r[k][1]=x;  
    r[k][2]=y;  
    //将走过的点标记为1防止死循环  
    a[x][y]=1;  
    //判断xy如果是终点，就打印路径  
    if(x==n&&y==n){  
        print(k);  
    }  
    int tx , ty;  
    for(int i=1;i<=4;i++){  
        tx=x+fx[i];  
        ty=y+fy[i];  
        if(a[tx][ty]=='0'){  
            dfs(tx,ty,k+1);  
        }  
    }  
}
```

```

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

```

**解法二：**通过数组标记走过的点，从而计算下一个点

**示例代码：**

```

#include<bits/stdc++.h>
using namespace std;
int n;
int r[400][3];//记录路径
char s[30][30];//字符数组，记录迷宫
int fx[5]={0,0,-1,0,1};
int fy[5]={0,-1,0,1,0};
//打印路径
void show(int k){
    int i;
    for(i=1;i<=k;i++){
        cout<<"("<<r[i][1]<<","<<r[i][2]<<")";
        //如果不是最后一个点，就加-->
        if(i!=k){
            cout<<"-->";
        } else{
            cout<<endl;
        }
    }
}
void fun(int k){
    int tx,ty;
    for(int i=1;i<=4;i++){
        tx=r[k-1][1]+fx[i];
        ty=r[k-1][2]+fy[i];
        if(tx>=1&&tx<=n&&ty>=1&&ty<=n&&s[tx][ty]=='0'){
            r[k][1]=tx;
            r[k][2]=ty;
            s[tx][ty]='1';
            if(tx==n&&ty==n){
                show(k);
            } else{
                fun(k+1);
            }
        }
    }
}
int main(){

```

```

int i,j;
cin>>n;
for(i=1;i<=n;i++){
    for(j=1;j<=n;j++){
        cin>>s[i][j];
    }
}
r[1][1]=1;
r[1][2]=1;
s[1][1]='1';
fun(2);
return 0;
}

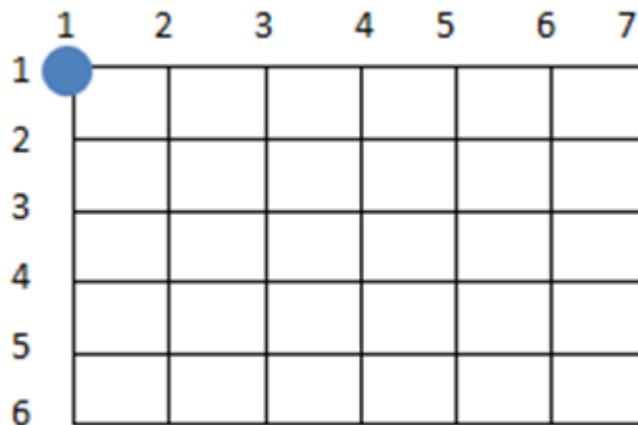
```

## 2.3 所有路径问题

### 【基础】卒的遍历

#### 题目描述

在一张 $n*m$ 的棋盘上（如6行7列）的最左上角（1,1）的位置有一个卒。该卒只能向下或者向右走，且卒采取的策略是先向下，下边走到头就向右，请问从（1,1）点走到（n,m）点可以怎样走，输出这些走法。



#### 输入

两个整数n, m代表棋盘大小 ( $3 \leq n \leq 8, 3 \leq m \leq 8$ )

#### 输出

卒的行走路线

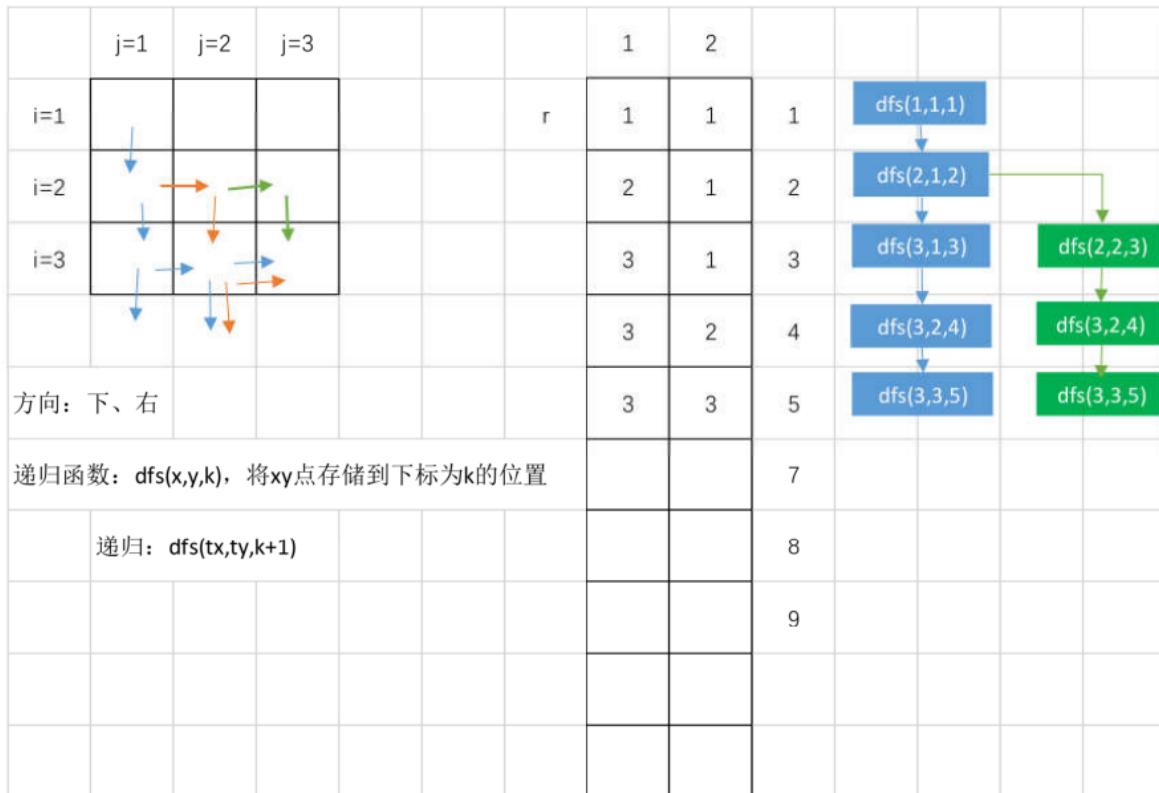
#### 示例输入

```
3 3
```

#### 示例输出

1:1,1->2,1->3,1->3,2->3,3  
2:1,1->2,1->2,2->3,2->3,3  
3:1,1->2,1->2,2->2,3->3,3  
4:1,1->1,2->2,2->3,2->3,3  
5:1,1->1,2->2,2->2,3->3,3  
6:1,1->1,2->1,3->2,3->3,3

**思路：**



**解法一：**参考迷宫的第一条路，深搜出迷宫的所有路径

```
#include<bits/stdc++.h>
using namespace std;
//只能向下或向右走，优先向下，其次向右
int m,n;
int r[20][3];//存储行走路径
//方向的变化
int fx[3]={0,1,0} ;
int fy[3]={0,0,1} ;
int c;//计数
void print(int k){
    c++;
    cout<<c<<" :";
    for(int i=1;i<=k;i++){
        cout<<r[i][1]<<","<<r[i][2]<<"-->" ;
    }
    cout<<n<<","<<m<<endl;
}
//向r数组下标为k的那一行记录x, y点
void dfs(int x,int y,int k){
    r[k][1]=x;
    r[k][2]=y;
```

```

//如果走到终点，打印路径
if(x==n&&y==m){
    print(k);
    //到达目标，无需递归
    return;
}
int tx,ty;
for(int i=1;i<=2;i++){
    tx=x+fx[i];
    ty=y+fy[i];
    //判断tx,ty是否有效
    if(tx>=1&&tx<=n&&ty>=1&&ty<=m) {
        dfs(tx,ty,k+1);
    }
}
int main(){
    cin>>n>>m;
    dfs(1,1,1);
    return 0;
}

```

**解法二：**利用数组存储上一个点的坐标，求解下一个点的坐标

```

#include<bits/stdc++.h>
using namespace std;
//只能向下或向右走，，优先向下，其次向右
int m,n;
int r[20][3];//存储行走路径
//方向的变化
int fx[3]={0,1,0} ;
int fy[3]={0,0,1} ;
int c;//计数
void print(int k){
    c++;
    cout<<c<<"";
    for(int i=1;i<=k;i++){
        cout<<r[i][1]<<","<<r[i][2]<<"-->";
    }
    cout<<n<<","<<m<<endl;
}
//向r数组下标为k的那一行记录x, y点
void dfs(int k){
    int tx,ty;
    for(int i=1;i<=2;i++){
        //得到新坐标
        tx=r[k-1][1]+fx[i];
        ty=r[k-1][2]+fy[i];
        //判断tx,ty是否有效
        if(tx>=1&&tx<=n&&ty>=1&&ty<=m) {
            //存储tx,ty
            r[k][1]=tx;
            r[k][2]=ty;
            //如果到了终点就打印，否则递归
        }
    }
}

```

```

        if(tx==n&&ty==m){
            print(k);
        }else{
            dfs(k+1);
        }
    }
}

int main(){
    cin>>n>>m;
    r[1][1]=1;
    r[1][2]=1;
    dfs(2);
    return 0;
}

```

## 2.4 课后作业

### 1. 【基础】走出迷宫的最少步数2

#### 题目描述

当你站在一个迷宫里的时候，往往会被错综复杂的道路弄得失去方向感，如果你能得到迷宫地图，事情就会变得非常简单。假设你已经得到了一个 $n*m$ 的迷宫的图纸，请你找出从起点到出口的最短路。

#### 输入

第一行是两个整数 $n$ 和 $m$ ( $1 \leq n, m \leq 100$ )，表示迷宫的行数和列数。接下来 $n$ 行，每行一个长为 $m$ 的字符串，表示整个迷宫的布局。字符'.'表示空地，'#'表示墙，'S'表示起点'T'表示出口。

#### 输出

输出从起点到出口最少需要走的步数。

#### 样例输入

```

3 3
S#T
.#
...

```

#### 样例输出

```
6
```

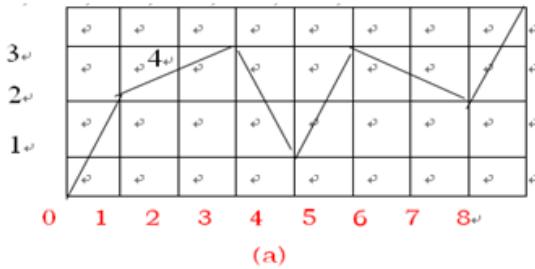
### 2. 【提高】马的遍历

#### 题目描述

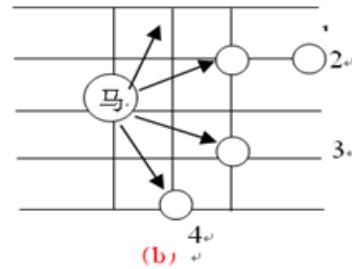
中国象棋半张棋盘如图 (a) 所示。马自左下角往右上角跳。今规定只许往右跳，不许往左跳，且要求马跳的方式按照 (b) 图顺时针深度优先递归。比如图 (a) 中所示为一种跳行路线。如果马要从 $0,0$ 点，跳到 $4,8$ 点，前6种跳法的打印格式如下，请参考前6种跳的方式，输出马从 $0,0$ 点到 $4,8$ 点所有可能的跳的路线。

1:0,0->2,1->4,2->3,4->4,6->2,7->4,8

2:0,0->2,1->4,2->3,4->1,5->3,6->4,8  
 3:0,0->2,1->4,2->3,4->1,5->2,7->4,8  
 4:0,0->2,1->4,2->2,3->4,4->3,6->4,8  
 5:0,0->2,1->4,2->2,3->4,4->2,5->4,6->2,7->4,8  
 6:0,0->2,1->4,2->2,3->4,4->2,5->0,6->2,7->4,8



(a)



(b)

### 输入

无

### 输出

按要求输出路径

## 3. 【基础】骑士巡游

### 题目描述

马在中国象棋以日字形规则移动，给定 $n \times m$ 大小的棋盘，以及马的初始位置 $(x, y)$ 和目标位置 $(s, t)$ ，要求不能重复经过棋盘上的同一个点，计算马至少走多少步可以到达目标位置，所有棋盘保证从初始位置到结束位置一定有路径可达。

### 输入

测试数据包含一行，为六个整数，分别为棋盘的大小以及初始位置坐标 $nmxyst$ 。 $(1 \leq x, s \leq n \leq 5, 1 \leq y, t \leq m \leq 5)$

### 输出

包含一行，为一个整数，表示马能到达目标位置的最小步数。

### 样例输入

```
3 3 1 1 1 3
```

### 样例输出

```
2
```

## 4. 【提高】小X学游泳

### 题目描述

小X想要学游泳。

这天，小X来到了游泳池，发现游泳池可以用 $N$ 行 $M$ 列的格子来表示，每个格子的面积都是1，且格子内水深相同。

由于小X刚刚入门，他只能在水深相同的地方游泳。为此，他把整个游泳池看成若干片区域，如果两个格子相邻（上下左右四个方向）且水深相同，他就认为它们属于同一片区域。

小X想知道最大的一片区域面积是多少，希望你帮帮他。

### 输入

第一行包含用一个空格隔开的两个整数N,M。 (1≤N,M≤100)  
接下来N行，每行包含M个 1到9的数字，表示每个格子的水深

### 输出

第一行包含一个整数，表示最大的一片区域面积。

### 样例输入

```
3 3
124
224
152
```

### 样例输出

```
3
```

## 5. 【提高】小X学游泳(swim)

### 题目描述

暑假快到啦，小X准备趁着这个暑假去学游泳。可是一开始小X就遇到了一个难题。  
游泳池划分成了一个  $n \times m \times n \times m$  的方格，这里  $n \times m \times n \times m$  表示  $n$  行  $m$  列。因为游泳池里的水深浅不一，所以这  $n \times m \times n \times m$  个方格对于小X的危险系数也会不一样。  
而小X目前需要从左上角的方格 (1, 1) 出发，游到右下角的方格 (n, m)，小X每次只能从当前方格游到上下左右四个相邻的方格中的某一格，并且在到达终点前不能离开游泳池。  
小X很担心会发生什么危险，所以希望你能帮他找一条危险系数最小的路径。  
一条路径的危险系数定义为这条路径所经过的方格的危险系数之和。  
注意：这条路径不能经过同一个方格两次（小X当然不希望去那么危险的地方再游一次）

### 输入

输入数据第一行有两个用空格隔开的正整数  $n$  和  $m$ ，表示泳池的行数和列数。  
接下来共有  $n$  行数据，每行有  $m$  个用空格隔开的大于等于 0 的整数，表示每个方格的危险系数

### 输出

输出仅有一行包含一个整数ans，表示要求的从左上角的方格 (1, 1) 出发，游到右下角的方格 (n, m) 的最小的危险系数。

### 样例输入

```
4 5
1 7 2 8 2
3 10 1 5 1
2 8 3 7 1
1 2 1 20 1
```

### 样例输出

```
19
```

### 提示

### 【数据范围】

对于 30% 的数据,  $1 \leq n, m \leq 51 \leq n, m \leq 51 \leq n, m \leq 5$

对于另外 40% 的数据,  $1 \leq n, m \leq 201 \leq n, m \leq 201 \leq n, m \leq 20$ , 每个方格的危险系数 = 0 或 1

对于 100% 的数据,  $1 \leq n, m \leq 301 \leq n, m \leq 301 \leq n, m \leq 30$ ,  $0 \leq$  每个方格的危险系数  $\leq 100000$

【样例解释】路径:  $(1, 1) \rightarrow (1, 2) \rightarrow (1, 3) \rightarrow (2, 3) \rightarrow (2, 4) \rightarrow (2, 5)$

$\rightarrow (3, 5) \rightarrow (4, 5)$ , 危险系数之和为  $1+7+2+1+5+1+1=19$ 。

### 【来源】

常州市2016“信息与未来”夏令营选拔赛

### 【建议】

请使用深搜和广搜分别实现

## 3. 回溯

### 3.1 回溯的概念

回溯是搜索算法中的一种控制策略, 也被称作“试探法”。它的基本思想是: 为了求得问题的解, 先选择一种可能的情况向前探索, 在探索的过程中, 一旦发现原来的选择是错误的, 就退回上一步重新选择, 继续向前探索, 如此反复, 直到得到解或者证明无解。本质上是对深搜的优化。

如迷宫问题, 进入迷宫后先随意选择一个方向一步步向前探索前进, 如果碰到死胡同说明前方已无路可走, 这是, 首先看其它方向是否还有路可走, 如果有路可走则沿着该方向再向前探索, 如果同样无路可走则退回一步看其它方向是否还有路可走, 如果有路可走就继续探索, 如果无路可走, 则按照此原则不断探索, 直到找到出路或从原路返回证明此题无解。

#### 实现思路1:

```
int search(int k){  
    for(i=1;i<=算法总数;i++){  
        保存结果(现场状态):  
        if(到达目的地){  
            输出解  
        }else{  
            search(k+1);  
        }  
        恢复: 保存结果之前的状态, 回溯一步;  
    }  
}
```

#### 实现思路2:

```
int search(int k){  
    if(到达目的地){  
        输出解  
    }else{  
        for(i=1;i<=算法总数;i++){  
            if(满足条件){  
                保存结果(现场状态);  
                search(k+1);  
                恢复: 保存结果之前的状态, 回溯一步;  
            }  
        }  
    }  
}
```

## 3.2 课堂案例

### 1. 【基础】迷宫的所有路径

#### 题目描述

已知一N×N的迷宫，允许往上、下、左、右四个方向行走，且迷宫中没有任何障碍，所有的点都可以走。现请你按照右、下、左、上顺序进行搜索，找出从左上角到右下角的所有路径。

#### 输入

输入一个整数N (N<=5) 代表迷宫的大小。

#### 输出

按右、下、左、上搜索顺序探索迷宫，输出从左上角1,1点走到右下角N,N点的所有可能的路径。

#### 样例输入

```
3
```

#### 样例输出

```
1:1,1->1,2->1,3->2,3->3,3  
2:1,1->1,2->1,3->2,3->2,2->3,2->3,3  
3:1,1->1,2->1,3->2,3->2,2->2,1->3,1->3,2->3,3  
4:1,1->1,2->2,2->2,3->3,3  
5:1,1->1,2->2,2->3,2->3,3  
6:1,1->1,2->2,2->2,1->3,1->3,2->3,3  
7:1,1->2,1->2,2->2,3->3,3  
8:1,1->2,1->2,2->3,2->3,3  
9:1,1->2,1->2,2->1,2->1,3->2,3->3,3  
10:1,1->2,1->3,1->3,2->3,3  
11:1,1->2,1->3,1->3,2->2,2->2,3->3,3  
12:1,1->2,1->3,1->3,2->2,2->1,2->1,3->2,3->3,3
```

#### 解法一：

```
#include<bits/stdc++.h>  
using namespace std;  
//右下左上  
/*  
思路：沿着右下左上的顺序深搜，走过的点标记为true  
递归其他方向，递归结束后退到上一步，撤销标记，回溯到前一个状态  
*/  
int n,c;  
int r[30][3];//存储路径  
bool f[10][10];//标记走过的点  
int fx[5]={0,0,1,0,-1};  
int fy[5]={0,1,0,-1,0};  
//输出路径  
void print(int k){  
    c++;  
    cout<<c<<" : "  
    for(int i=1;i<k;i++){  
        cout<<r[i][1]<<"," <<r[i][2]<<"->"  
    }  
    cout<<n<<" , "<<n<<endl;
```

```

}

//深搜存储路径，将xy点存储到r数组下标为k的位置
void dfs(int x,int y,int k){
    r[k][1]=x;
    r[k][2]=y;
    //如果到达终点就打印
    if(x==n&&y==n){
        print(k);
        return ;
    }
    //尝试不同方向
    int tx,ty;
    for(int i=1;i<=4;i++){
        tx=x+fx[i];
        ty=y+fy[i];
        if(tx>=1&&tx<=n&&ty>=1&&ty<=n&&f[tx][ty]==false){
            //标记txty点走过
            f[tx][ty]=true;
            dfs(tx,ty,k+1);
            //递归结束后退，回溯到前一个状态
            f[tx][ty]=false;
        }
    }
}
int main(){
    cin>>n;
    //从1,1点开始递归，记录到r数组下标为1的那一行
    f[1][1]=true;//标记1,1走过
    dfs(1,1,1);
    return 0;
}

```

## 解法二：

```

#include<bits/stdc++.h>
using namespace std;
//右下左上
/*
思路：沿着右下左上的顺序深搜，走过的点标记为true
递归其他方向，递归结束后退到上一步，撤销标记，回溯到前一个状态
*/
int n,c;
int r[30][3];//存储路径
bool f[10][10];//标记走过的点
int fx[5]={0,0,1,0,-1};
int fy[5]={0,1,0,-1,0};
//输出路径
void print(int k){
    c++;
    cout<<c<<" :";
    for(int i=1;i<k;i++){
        cout<<r[i][1]<<"," <<r[i][2]<<"->";
    }
    cout<<n<<"," <<n<<endl;
}

```

```

//深搜存储路径，将xy点存储到r数组下标为k的位置
void dfs(int k){
    int tx,ty;
    for(int i=1;i<=4;i++){
        tx=r[k-1][1]+fx[i];
        ty=r[k-1][2]+fy[i];
        if(tx>=1&&tx<=n&&ty>=1&&ty<=n&&f[tx][ty]==false){
            //存储路径
            r[k][1]=tx;
            r[k][2]=ty;
            //标记tx,ty走过
            f[tx][ty]=true;
            if(tx==n&&ty==n){
                print(k);
            }else{
                dfs(k+1);
            }
            //递归结束，回溯到上一个状态
            f[tx][ty]=false;
        }
    }
}

int main(){
    cin>>n;
    r[1][1]=1;
    r[1][2]=1;
    f[1][1]=true;

    dfs(2);
    return 0;
}

```

## 2. 【基础】全排列的结果

### 题目描述

从键盘读入一个整数n ( $n \leq 6$ )，请输出1~n中所有整数的全排列，按照由小到大输出结果，每组的n个数之间用空格隔开。

全排列的含义：从n个不同元素中任取m ( $m \leq n$ ) 个元素，按照一定的顺序排列起来，叫做从n个不同元素中取出m个元素的一个排列。当m=n时所有的排列情况叫全排列。

如当n=3时，全排列的结果为：

```

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

```

### 输入

一个整数n ( $n \geq 1 \&\& n \leq 6$ )

### 输出

1~n中所有数的全排列的结果，按照由小到大输出，每行n个数

### 样例输入

## 样例输出

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

## 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int n,a[10];//存放全排列的结果
bool f[10];//标记哪些数使用过
void print(){
    for(int i=1;i<=n;i++){
        cout<<a[i];
        if(i!=n){
            cout<<" ";
        }else{
            cout<<endl;
        }
    }
}
//递归函数，为a数组的每个元素赋值
void fun(int k){
    for(int i=1;i<=n;i++){
        //如果i没有别用过，就填写到下标为k的位置
        if(!f[i]){
            a[k]=i;
            //标记i已经被用过
            f[i]=true;
            //如果a中存储了n个元素，输出结果，否则递归为k+1的下标赋值
            if(k==n){
                print();
            } else{
                fun(k+1);
            }
            //回溯到上一个状态，标记i未被使用
            f[i]=false;
        }
    }
}
int main(){
    cin>>n;
    //为a数组的下标为1的位置赋值
    fun(1);
    f[1]=1;
    return 0;
}
```

### 3. 【提高】素数环

#### 题目描述

从1~n ( $2 \leq n \leq 10$ ) 这n个数，摆成一个环，要求相邻的两个数的和是素数，按照由小到大请输出所有可能的摆放形式。

比如：n = 4，输出形式如下

1:1 2 3 4

2:1 4 3 2

3:2 1 4 3

4:2 3 4 1

5:3 2 1 4

6:3 4 1 2

7:4 1 2 3

8:4 3 2 1

total:8

比如：n = 6，输出形式如下

1:1 4 3 2 5 6

2:1 6 5 2 3 4

3:2 3 4 1 6 5

4:2 5 6 1 4 3

5:3 2 5 6 1 4

6:3 4 1 6 5 2

7:4 1 6 5 2 3

8:4 3 2 5 6 1

9:5 2 3 4 1 6

10:5 6 1 4 3 2

11:6 1 4 3 2 5

12:6 5 2 3 4 1

total:12

#### 输入

一个整数n ( $2 \leq n \leq 10$ )

#### 输出

前若干行，每行输出一个素数环的解，最后一行，输出解的总数

#### 样例输入

4

#### 样例输出

```
1:1 2 3 4  
2:1 4 3 2  
3:2 1 4 3  
4:2 3 4 1  
5:3 2 1 4  
6:3 4 1 2  
7:4 1 2 3  
8:4 3 2 1  
total:8
```

### 思路：

1. 数据初始化
2. 递归填数，判断dii个数字填入是否合法
  1. 如果合法就填入，判断是否达到目标，是就打印结果，不是就递归下一个
  2. 不合法就选择下一种可能

### 示例代码：

```
#include<bits/stdc++.h>  
using namespace std;  
int a[20];//存放选数的结果  
bool f[20];//标记数字是否被用过  
int n,c;  
bool judge(int n){  
    if(n==1) return false;  
    for(int i=2;i<=sqrt(n);i++){  
        if(n%i==0){  
            return false;  
        }  
    }  
    return true;  
}  
void print(){  
    c++;  
    cout<<c<<" : ";  
    for(int i=1;i<=n;i++){  
        cout<<a[i];  
        if(i!=n){  
            cout<<" ";  
        }else{  
            cout<<endl;  
        }  
    }  
}  
//递归函数  
void fun(int k){  
    //下标为k的位置有n种选法  
    for(int i=1;i<=n;i++){  
        //如果i未被选过且i和前一个数的和是素数  
        if(!f[i]&&(k==1||judge(i+a[k-1]))){  
            a[k]=i;  
            f[i]=true;  
            if(k==n&&judge(a[k]+a[1])){  
                print();  
            }  
            fun(k+1);  
            f[i]=false;  
            a[k]=0;  
        }  
    }  
}
```

```

        print();
    } else{
        fun(k+1);
    }
    //回溯，撤销i用过的标记
    f[i]=false;
}
}
int main(){
cin>>n;
fun(1);
cout<<"total:"<<c;
return 0;
}

```

#### 4. 【提高】素数分解

##### 题目描述

素数，又称质数，是指除 1 和其自身之外，没有其他约数的正整数。例如 2、3、5、13 都是素数，而 4、9、12、18 则不是。

虽然素数不能分解成除 1 和其自身之外整数的乘积，但却可以分解成更多素数的和。

你需要编程 求出一个正整数最多能分解成多少个互不相同的素数的和。

例如， $21 = 2 + 19$ 是21的合法分解方法。 $21 = 2 + 3 + 5 + 11$ 则是分解为最多素数的方法。

再比如：128，最多可以分解为9个素数的和。

输入

$n$  ( $10 \leq n \leq 200$ )。

输出

$n$  最多能分解成多少个不同的素数的和。

样例输入

21

样例输出

4

示例代码：

```

#include<bits/stdc++.h>
using namespace std;
int num[50];
int n;
int index;
int maxtotal=-1;
//素数判断函数
bool judge(int n){
    if(n==1) return false;
    for(int i=2;i<=sqrt(n);i++){

```

```

        if(n%i==0){
            return false;
        }
    }
    return true;
}

//k为当前下标, sum为综合, total为使用的数字个数
void fun(int k,int sum,int total){
    if(sum==n){
        if(total>maxtotal){//更新total
            maxtotal=total;
        }
    }else if(sum<n&&k<index){//如果sum超过n, 或者下标大于素数个数就结束
        fun(k+1,sum+num[k],total+1);
        fun(k+1,sum,total);
    }
}
int main(){
    cin>>n;
    for(int i=2;i<=n;i++){
        if(judge(i)){
            num[index++]=i;
        }
    }
    fun(0,0,0);
    cout<<maxtotal;
    return 0;
}

```

### 3.3 课后作业

#### 1. 【提高】迷宫的路径?

##### 题目描述

Mitch老鼠在森林里游玩，不小心走进了一个迷宫里面，这个迷宫是一个n行m列的矩阵，迷宫中有些格子是可以走的，有些格子是不能走的，能走的格子用“o”（小写字母o）表示，不能走的格子用“#”表示。

Mitch选择走出迷宫的策略是：先向右，如果右边走不通则选择向下，如果下边走不通则选择向左，如果左边走不通则选择向上；如果四个方向都走不通，则后退选择其他能走的路径。

Mitch从类似下图所示的迷宫的左上角（1,1）点进入迷宫（请注意：入口1,1和出口的n,m点都不是#），请问Mitch有哪些方法可以走出迷宫，走到（n,m）点；请编程求出所有可能的路径，输出这些路径，如果不存在任何的路径可以走出迷宫，请输出“no”。

	1	2	3	4	5
1	o	o	o	o	o
2	o	#	#	#	#
3	o	o	o	o	o
4	#	o	o	#	o
5	o	o	o	o	#
6	o	#	o	o	o

## 输入

第一行包含两个整数n和m，中间用单个空格隔开，代表迷宫的行和列的数量。

接下来n行，每行m个字符，描述迷宫地图。字符只有“o”或“#”两种，“o”代表这个格子可以走，“#”代表这个格子不能走。(4 <= n,m <= 10 )

## 输出

请按照Mitch选择的走出迷宫的策略，输出所有可能的路径，输出形式请参考样例输出的形式。

### 样例输入

```
6 5
ooooo
o#####
ooooo
#oo#o
oooo#
o#ooo
```

### 样例输出

```
1:1,1->2,1->3,1->3,2->3,3->4,3->5,3->5,4->6,4->6,5
2:1,1->2,1->3,1->3,2->3,3->4,3->5,3->6,3->6,4->6,5
3:1,1->2,1->3,1->3,2->3,3->4,3->4,2->5,2->5,3->5,4->6,4->6,5
4:1,1->2,1->3,1->3,2->3,3->4,3->4,2->5,2->5,3->6,3->6,4->6,5
5:1,1->2,1->3,1->3,2->4,2->4,3->5,3->5,4->6,4->6,5
6:1,1->2,1->3,1->3,2->4,2->4,3->5,3->6,3->6,4->6,5
7:1,1->2,1->3,1->3,2->4,2->5,2->5,3->5,4->6,4->6,5
8:1,1->2,1->3,1->3,2->4,2->5,2->5,3->6,3->6,4->6,5
```

## 2. 【提高】方格取数

### 题目描述

在 n 行、 m 列的方格矩阵中，每个方格都包含一个数字。小明可以从任意方格出发开始移动。每次移动可以移到与当前方格有一条边相邻的方格(即向上、下、左或右方向移动 1 格，且不能移出边界)。除此之外，你移动到的方格中的数字必须比当前方格中的数字更大。

请你帮助小明编程规划移动路径，使路径上经过的所有数字之和最大。

本题方格中的数据根据输入的初始数字 s 按照如下算法生成:

```
for i = 1, 2, ... n
    for j = 1, 2, ... m
        s ← (s × 345) mod 19997
```

矩阵第 i 行第 j 列方格中的数字为(s mod 10) + 1

## 输入

正整数 n, m (方格的大小), s (数据生成器的初始数值)。 $1 \leq n, m \leq 100$ ,  $1 \leq s \leq 19,997$ 。

## 输出

所有合法路径中的最大数字和。

### 样例输入

```
4 5 97
```

### 样例输出

**样例输入**

40 50 1

**样例输出**

47

**3. 【基础】n个数取出r个数排列****题目描述**

从1~n任意挑出r个数进行排列，请从小到大输出所有可能的排列结果。

如：n=5，r=2，则输出结果如下

1 2

1 3

1 4

1 5

2 1

2 3

2 4

2 5

3 1

3 2

3 4

3 5

4 1

4 2

4 3

4 5

5 1

5 2

5 3

5 4

**输入**

两个整数n和r（n和r都是3~6之间的整数）

**输出**

从1~n中去r个数的排列结果！

**样例输入**

5 2

**样例输出**

1 2

1 3

1 4

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

#### 4. 【提高】素数环2

##### 题目描述

将 $1 \sim n$ 这 $n$ 个数字首尾相连，形成一个圆环，要求圆环上任意两个相邻的数字之和都是一个素数，请编程输出符合条件的素数环。

##### 输入

输入数据仅一行，包含一个正整数 $n$  ( $n \leq 20$ )。

##### 输出

输出数据最多包括10行，每行由 $n$ 个整数组成，表示前十个符合条件的素数环（不足十个时全部输出）。所有素数环第一个元素必须是1，且按照从小到大的顺序排列。

##### 样例输入

```
6
```

##### 样例输出

```
1 4 3 2 5 6  
1 6 5 2 3 4
```

#### 5. 【基础】简单单词接龙

##### 题目描述

有 $n$ 个单词 ( $1 \leq n \leq 50$ )，每个单词由2个小写字母组成，并约定第1个单词为龙头。

例如： $n=7$

7个单词为

aa  
ac  
ab  
de  
bh  
hk  
cd

接龙的方法为前一个单词的第2个字母和后一个单词的第一个字符相同，此时，可接的方法有：

aa—ac—cd 长度为3，即龙上有3个单词

aa—cd-de 长度也为3

aa—ab—bh—hk 长度为4

程序要求给出单词之后，求出最长龙的长度。

### 输入

第一行一个整数n

接下来n行，每行2个字母表示一个单词（单词字母间无空格）

### 输出

输出一个整数，即最长的接长度（即龙上单词的个数）。

### 样例输入

```
7
```

```
aa
```

```
ac
```

```
ab
```

```
de
```

```
bh
```

```
hk
```

```
cd
```

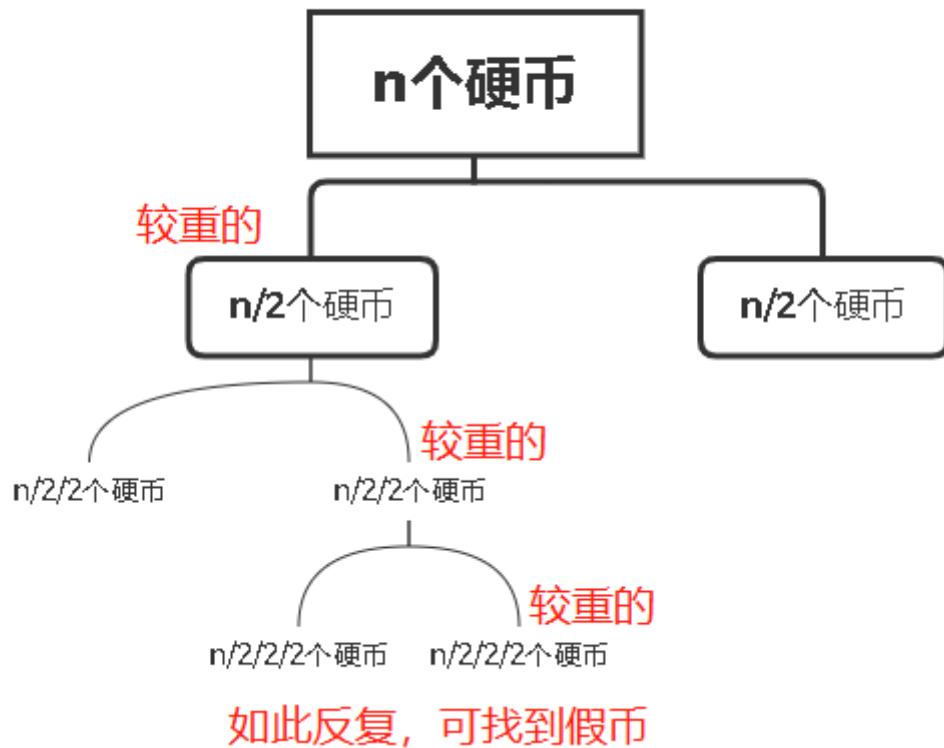
### 样例输出

```
4
```

## 第三十章 分治与排序

### 1.什么是分治？

**问题：**在若干个硬币中混入了一枚假币，外观与真币相同，但由于材质不同，所以假币要轻一些，请你用提供的天平用最快的方法将假硬币找出来。



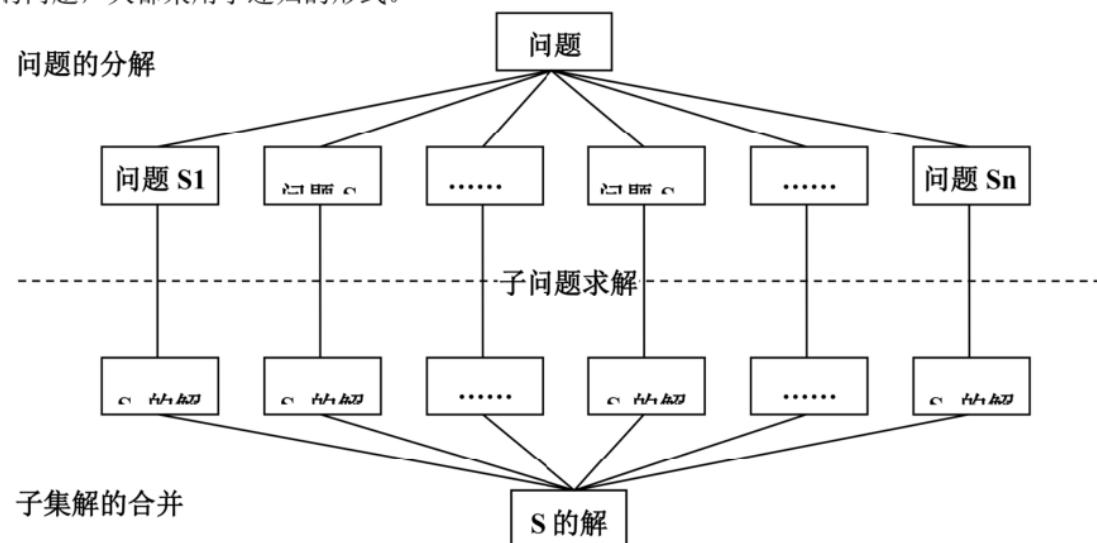
**分治：**将一个规模为n的问题分解为k个规模较小的子问题，这些子问题互相独立且与原问题性质相同。求出子问题的解就可以得到原问题的解。即分治是一种分目标完成任务的程序算法。分治的一般步骤：

1. 分解：将要解决的问题划分为若干个规模较小的同类问题；
2. 求解：当子问题划分足够小时，用较简单的方法解决；
3. 合并：按原问题的要求，将子问题的解合并为原问题的解

运用分治策略解决问题一般来说具有以下特点：

1. 原问题可以分解为多个子问题  
这些子问题与原问题相比，只是问题的规模有所降低，其结构和求解方法与原问题相同或相似。
2. 原问题在分解过程中，递归地求解子问题  
由于递归都必须要有一个终止条件，因此吗，当分解后的子问题规模足够小时，应该可以直接求解。
3. 在求解并得到各个子问题的解后，应该能够采用某种方式方法合并或构造出原问题的解。

不难发现，在分治策略中，由于子问题与原问题在结构上的相似性，用分值方法解决的问题大都采用了递归的形式



## 2. 分治的应用

### 1. 【入门】二分查找

#### 题目描述

请在一个有序递增数组中（不存在相同元素），采用二分查找，找出值x的位置，如果x在数组中不存在，请输出-1！

#### 输入

第一行，一个整数n，代表数组元素个数 ( $n \leq 106$ )

第二行，n个数，代表数组的n个递增元素 ( $1 \leq \text{数组元素值} \leq 108$ )

第三行，一个整数x，代表要查找的数 ( $0 \leq x \leq 108$ )

#### 输出

x在数组中的位置，或者-1。

#### 样例输入

```
10
1 3 5 7 9 11 13 15 17 19
3
```

#### 样例输出

```
2
```

#### 二分查找非递归解法：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n,a[10000]={0},x,r,i,mid,left,right;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    cin>>x;
    left=0;
```

```

right=n-1;
r=-1;//假设没找到
while(left<=right){
    mid=(left+right)/2;
    if(a[mid]==x){
        r=mid;
        break;
    }else if(x<a[mid]){
        right=mid-1;
    }else if(x>a[mid]){
        left = mid +1;
    }
}
cout<<(r== -1?r:r+1);
return 0;
}

```

## 二分查找递归解法：

```

#include<bits/stdc++.h>
using namespace std;
int a[10000],x;
int fun(int left,int right){
    if(left<=right){
        int mid = (left+right)/2;
//        cout<<a[mid]<<endl;
        if(x<a[mid]){
            return fun(left,mid-1);
        }else if(x>a[mid]){
            return fun(mid+1,right);
        }else{
            return mid;
        }
    }else{
        return -1;
    }
}
int main(){
    int n,r,i;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    cin>>x;
    r = fun(0,n-1);
    cout<<(r== -1?r:r+1);
    return 0;
}

```

## 插值查找解法：

适用场景：数据量较大且关键字分布比较均匀的查找表

原理：英汉字典中查找nice时我们肯定不会仿照对半查找，而是在nice附近开始查找。

**关键代码：int mid =first+(x-a[first])\*((last-first)/(a[last]-a[first]));计算出可能的下标**

```
#include<bits/stdc++.h>
using namespace std;
int a[10000],x;
int fun(int left,int right){
    if(left<=right){
        int mid = left+(x-a[left])*((right-left)/(a[right]-a[left]));
        if(x<a[mid]){
            return fun(left,mid-1);
        }else if(x>a[mid]){
            return fun(mid+1,right);
        }else{
            return mid;
        }
    }else{
        return -1;
    }
}
int main(){
    int n,r,i;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    cin>>x;
    r = fun(0,n-1);
    cout<<(r== -1?r:r+1);
    return 0;
}
```

## 2. 【入门】数组元素的排序

### 题目描述

对数组的元素按从小到大进行排序。

### 输入

有两行 第一行有一个整数n( 5 <= n <= 10 ) 第二行有n个整数

### 输出

输出更新后的数组

### 样例输入

```
8
1 2 3 6 8 7 4 5
```

### 样例输出

```
1 2 3 4 5 6 7 8
```

**快速排序法：**

快速排序首先选择一个轴值 (pivot, 也叫基准值) , 将待排序记录划分成独立的两个部分, 左侧的元素均小于轴值, 右侧的元素均大于或等于轴值, 然后对着两部分再重复, 直到整个序列有序。过程类似二叉搜索树, 就是一个递归的过程。

```
#include<bits/stdc++.h>
using namespace std;
int a[1000],n;
int quick(int left,int right){
    /*
    选一个轴值, 使得左侧的数<轴值, 右侧的数>=轴值
    */
    int i,j,mid=(left+right)/2;
    i=left;
    j=right;
    //当i<=j时, 找mid左侧>=a[mid]的数, mid右侧<=a[mid]的数, 交换他们
    while(i<=j){
        while(a[i]<a[mid]) i++;
        while(a[j]>a[mid]) j--;
        if(i<=j){
            swap(a[i],a[j]);
            i++;
            j--;
        }
    }
    //采用分治的思想递归
    if(left<=j) quick(left,j);
    if(i<=right) quick(i,right);
}
int main(){
    int i;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    quick(0,n-1);
    for(i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    return 0;
}
```

### 3.常见的排序算法

#### 3.1 冒泡排序

##### 【入门】数组元素的排序

###### 题目描述

对数组的元素按从小到大进行排序。

###### 输入

有两行 第一行有一个整数n( 5 <= n <= 10 ) 第二行有n个整数

输出

输出更新后的数组

### 样例输入

```
8  
1 2 3 6 8 7 4 5
```

### 样例输出

```
1 2 3 4 5 6 7 8
```

### 示例代码：

```
#include<bits/stdc++.h>  
using namespace std;  
int main(){  
    int a[10000],n,i,j;  
    cin>>n;  
    for(i=0;i<n;i++){  
        cin>>a[i];  
    }  
    for(i=0;i<n;i++){  
        for(j=0;j<n-i-1;j++){  
            if(a[j]>a[j+1]){  
                swap(a[j],a[j+1]);  
            }  
        }  
    }  
    for(i=0;i<n;i++){  
        cout<<a[i]<<" ";  
    }  
    return 0;  
}
```

n个数最多产生 $1+2+3+\dots+n-1$ 轮循环

## 3.2 冒泡排序Pro

改进思路：如果第i轮循环结束没有产生任何交换，说明第i轮结束后数组已经有序，则无需进行下一次循环

### 示例代码：

```
#include<bits/stdc++.h>  
using namespace std;  
int main(){  
    int a[10000],n,i,j;  
    bool f=false;//标记数组是否产生过交换  
    cin>>n;  
    for(i=0;i<n;i++){  
        cin>>a[i];  
    }
```

```

for(i=0;i<n;i++){
    for(j=0;j<n-i-1;j++){
        if(a[j]>a[j+1]){
            swap(a[j],a[j+1]);
            f=true;
        }
    }
    if(!f) break;
}
for(i=0;i<n;i++){
    cout<<a[i]<<" ";
}
return 0;
}

```

### 3.3 选择排序

5 4 3 2 1

第一轮：

4 5 3 2 1

4 3 5 2 1

4 3 2 1 5

问题：通过交换让最大数到最后，会产生很多次的交换，可以优化为：找到最大数的下标，如果最大数的下标不是该轮的最后一个数的下标，直接交换最大数下标和最后一个数下标对应的数。

选择排序：

第一轮：最大数下标是4，第一个数下标是0：

1 4 3 2 5

第二轮：最大数下标4，最后一个数下标3：

1 2 3 4 5

第三轮：最大数下标2，最后一个数下标2

...

第*i*轮最后一个数的下标：n-i

冒泡排序第*i*轮是将本轮的最大数通过不断比较的方式移动到本轮的最后一个位置；

选择排序：n个数比较n-1轮，找到第*i*轮的最大数下标m，判断如果不是本轮最后一个数的下标n-i，那就交换a[m]和a[n-i]；

选择排序相较于冒泡排序，减少了交换的次数。

```

#include<bits/stdc++.h>
using namespace std;
int a[100],n,i,j,m,k;
int main(){
    cin>>n ;

```

```

for(i=0;i<n;i++){
    cin>>a[i];
}
//n个数需要比较n-1轮
for(i=0;i<n-1;i++){
    //假设本轮中下标为0的数是最大数
    m=0;
    for(j=0;j<n-i;j++){
        if(a[j]>a[m]){
            m=j;
        }
    }
    //如果第i轮最大数下标不是本轮最后一个数下标(n-1)
    if(m!=n-i-1){
        swap(a[m],a[n-i-1]);
    }
}
for(i=0;i<n;i++){
    cout<<a[i]<<" ";
}
return 0;
}

```

### 3.4 桶排序

桶排序也叫箱排序。工作原理：统计每个元素出现的次数，然后利用桶的下标已经有序的原理，按照每个数出现的次数循环输出，因此，桶排序要求数组元素数值必须是小范围内的数。

比如有1000000个数需要排序，每个数的数值都在1~1000的范围内，就可以使用桶排序。

如果数据量不大但是某一个数据特别大，就不适合。

简单来说，桶排序就是之前讲过的数组标记法。

i下标代表了排序的元素，a[i]代表i出现的次数

```

#include<bits/stdc++.h>
using namespace std;
/*数据范围为1~100*/
int a[101],n,i,j,k;
int main(){
    cin>>n ;
    for(i=0;i<n;i++){
        cin>>k;
        a[k]++;
    }
    for(i=1;i<=100;i++){
        for(j=0;j<a[i];j++){
            cout<<i<<" ";
        }
    }
    return 0;
}

```

### 3.5 插入排序

插入排序将数组分成前后两块，前一块有序区间，后一块无序区间。

排序思想：从无序区间取一个数字插入有序区间，如此有序区间扩大，无序区间缩小，直到有序区间覆盖整个数组。

无序区间：i 到 n-1；

有序区间：0 到 i-1

插入关键：我们需要找到有序区间中插入的位置j后面（注意：对有序区间的遍历一定要是倒序的，一旦发现有比a[i]小的数据a[j]，就插入到j后面的位置），然后将有序区间j后面的数字往后挪一位，将j+1的位置空出来，放入需要插入的数字。

```
#include<bits/stdc++.h>
using namespace std;
int a[1000], n, i, j, k, t;

int main(){
    cin >> n;
    for(i=0; i < n; i++){
        cin >> a[i];
    }
    // 循环无序区间，(a[0]默认在有序区间)
    for(i=1; i < n; i++){
        // 将a[i]插入有序区间，循环有序区间找到a[i]应该插入的下标
        for(j=0; j < i; j++){
            // 找到有序区间中的第一个比要插入的数字大的数，其位置就是要插入的位置
            if(a[j] >= a[i]){
                break;
            }
        }
        // 如果找到要插入的位置
        if(j != i) {
            t = a[i]; // 临时存起a[i]
            for(k=i-1; k >= j; k--) {
                a[k+1] = a[k];
            }
            a[j] = t;
        }
    }
    for(i=0; i < n; i++) {
        cout << a[i] << " ";
    }
    return 0;
}
```

## 3.6 快速排序

见课堂案例

```
#include<bits/stdc++.h>
using namespace std;
int a[1000],n;
int quick(int left,int right){
    /*
    选一个轴值，使得左侧的数<轴值，右侧的数>=轴值
    */
    int i,j,mid=(left+right)/2;
    i=left;
    j=right;
    //当i<=j时，找mid左侧>=a[mid]的数，mid右侧<=a[mid]的数，交换他们
    while(i<=j){
        while(a[i]<a[mid]) i++;
        while(a[j]>a[mid]) j--;
        if(i<=j){
            swap(a[i],a[j]);
            i++;
            j--;
        }
    }
    //采用分治的思想递归
    if(left<=j) quick(left,j);
    if(i<=right) quick(i,right);
}
int main(){
    int i;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    quick(0,n-1);
    for(i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    return 0;
}
```

## 3.7 归并排序

归并排序主要分为两个部分：

1.划分子区间

2.合并子区间

```
#include<bits/stdc++.h>
using namespace std;
void Merge(int sourceArr[],int tempArr[],int starIndex,int midIndex,int endIndex)
{
```

```

int i = starIndex;//归并时左边区域的起始位置
int j = midIndex+1;//归并时右边区域的起始位置
int k = starIndex;//临时区域的起始位置
//i:左边区域的起始位置没有移动到该区域的结束位置 midIndex+1
//j:右边区域的起始位置没有移动到该区域的结束位置 endIndex+1
while(i!=midIndex+1&&j!=endIndex+1){
    //分别比较两边区域的数字，将其中较小的复制到临时区tempArr中
    //复制完成后下标往后移动一位
    if(sourceArr[i]>sourceArr[j]){
        tempArr[k++]=sourceArr[j++];//右边区域的小，复制过来下标后移
    } else{
        tempArr[k++]=sourceArr[i++];//左边区域的小，复制过来下标后移
    }
}
//如果左边区域的数据没有复制完，继续将左边区域的数据复制到临时区域中

while(i!=midIndex+1){
    tempArr[k++]=sourceArr[i++];
}
while(j!=endIndex+1){
    tempArr[k++]=sourceArr[j++];
}
for(i=starIndex;i<=endIndex;i++){
    sourceArr[i]=tempArr[i];
}
}

//将数组分解成左右2块
void MergeSort(int sourceArr[],int tempArr[],int starIndex,int endIndex ){
    int midIndex;
    //分解到只有1个数的时候，不用继续分解
    if(starIndex<endIndex){
        //继续分解成左右两个区域
        midIndex=(starIndex+endIndex)/2;
        //分解左边区域
        MergeSort(sourceArr,tempArr,starIndex,midIndex);
        //分解右边区域
        MergeSort(sourceArr,tempArr,midIndex+1,endIndex);
        //归并操作
        Merge(sourceArr,tempArr,starIndex,midIndex,endIndex);
    }
}

int main(){
    int a[100];
    int i,b[100];
    int n;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    //将数组分解为2个部分
    MergeSort(a,b,0,n-1);
    cout<<"执行结束"<<endl;
    for(i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
}

```

```
    return 0;  
}
```

## 第三十一章 广度优先搜索 (BFS)

### 1. 什么是广度优先搜索

#### 1.1 基本概念

广度优先搜索算法 (Breadth First Search) , 又被称为“宽度优先搜索”或者“横向优先搜索”。简称BFS。

其思路为从图上的一个节点出发, 先访问其直接相连的子节点, 若子节点不符合, 再访问其子节点的子节点, 按级别顺序依次访问, 直到访问到目标节点。

#### 1.2 深搜与广搜的区别

##### 1.2.1 遍历方式

1. 深度优先DFS: 一个递归过程, 有回退的过程, 尽可能“深”地搜索地图。在深度优先搜索中, 对于最新发现的顶点, 如果它还有以此为起点而为探索到的边, 就沿此边继续搜索下去。当节点v的所有边都已被探寻过, 搜索将回溯到发现节点v有那条边的始节点, 则选择其中一个座位源节点并重复以上的过程, 整个进程反复进行直到所有节点都被发现为止。
2. 广度优先BFS: 一个分层的搜索过程, 没有回退的过程, 是非递归的。只是每次都尽可能地扩展当前节点的邻居节点, 之后再向其子节点进行扩展。

##### 1.2.2 应用上的区别

1. BFS: 对于解决最短或者最少问题特别有效 (因为BFS只要访问到某个点一定是最短路径到达的点) , 而且寻找深度小, 但缺点的内存耗费量大 (需要大量的数组单元存储状态)
2. DFS: 对于解决遍历和求所有问题有效, 对于问题搜索深度小的时候处理速度迅速, 然而在深度很大的情况下效率不高。

##### 1.2.3 实现时用到的数据结构

广搜使用队列 (queue) 来实现, 整个过程可以看成一个倒立的树形:

1. 把根节点放到队列的末尾
2. 每次从队列的头部取出一个元素, 查看这个元素所有的下一级元素, 把它们放到队列的末尾。并把这个元素记为它下一级元素的前驱。
3. 找到所要找的元素时结束程序。
4. 如果遍历整个树都没有找到, 结束程序。

### 2. 广搜的应用

#### 1.课堂练习

##### 1. 【入门】快乐的马里奥

###### 1. 题目描述

马里奥是一个快乐的油漆工人，这天他接到了一个油漆任务，要求马里奥把一个n行m列的矩阵每一格都用油漆标记一个数字，标记的顺序按照广度优先搜索的方式进行，也就是他会按照如下方式标记：

- 1、首先标记第1行第1列的单元格，标记数字为1；
  - 2、然后标记当前单元格上下左右四个方向所有能标记的单元格，且：
    - ① 标记顺序按照：右、下、左、上的优先级；
    - ② 不能标记到矩阵外，且标记过的数字不能重复标记；
  - 3、当本单元格标记结束，寻找比本单元格中数字大1的单元格，标记那个单元格的上下左右四个方向，也是按照步骤2所示的要求进行标记。
- 依次类推，直到所有单元格都被标记。

比如：如果有一个 $3 * 3$ 的矩阵如下，那么首先标记1,1单元格，并按照上面步骤2的要求标记其四周能够标记的单元格，标记结果如下：

	1	2	3
1	1	2	
2	3		
3			

接下来，标记比1,1格大1的数字的四周的单元格，也就是标记值为2的单元格四周的单元格，标记结果如下：

	1	2	3
1	1	2	4
2	3	5	
3			

接下来标记值为3的单元格四周的单元格，标记结果如下：

	1	2	3
1	1	2	4
2	3	5	
3	6		

接下来标记值为4的单元格四周的单元格，标记结果如下：

	1	2	3
1	1	2	4
2	3	5	7
3	6		

接下来标记值为5的单元格四周的单元格，标记结果如下：

	1	2	3
1	1	2	4
2	3	5	7
3	6	8	

接下来标记值为6的单元格四周的单元格，但这个数字四周的单元格已经被标记，因此继续标记值为7四周的单元格，标记结果如下：

	1	2	3
1	1	2	4
2	3	5	7
3	6	8	9

此时，发现标记结束，得到如上图所示的标记结果。

### 输入

两个整数n和m，n和m都是3~100之间的整数。

### 输出

输出n行m列的标记后的矩阵，输出每个数后空一格。

### 样例输入

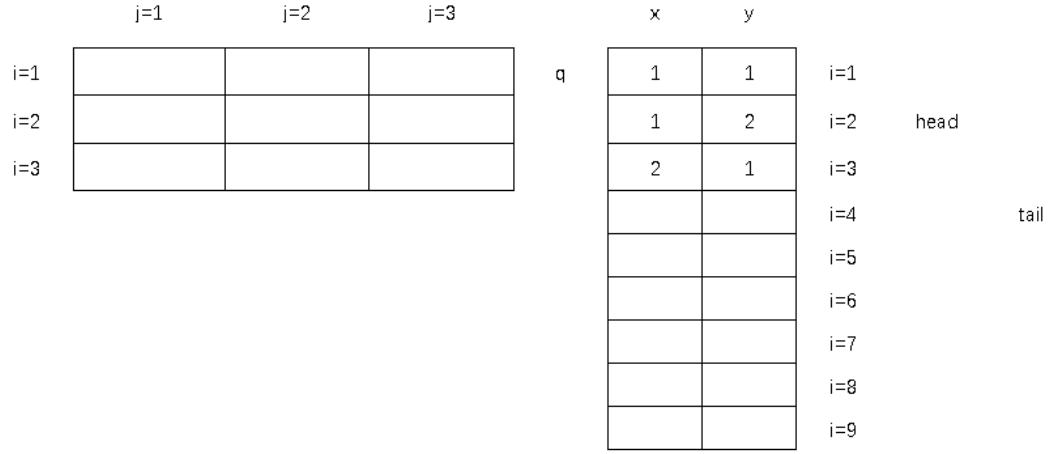
```
3 3
```

### 样例输出

```
1 2 4
3 5 7
6 8 9
```

### 思路：

用一个二维数组（队列queue，FIFO）存储广搜遍历的每个点，将head四个方向的可访问的点（没有出矩阵，且点没有访问过）遍历标记并将这些点存入队列，一个点四个方向都尝试访问结束后，head++，尝试新的点，直到head>tail，表示队列中每个点四个方向都标记结束了。



### 示例代码

```

#include<bits/stdc++.h>
using namespace std;
//a:存储矩阵, q: 存储队列 (遍历过的点)
int a[110][110],q[10100][3];
//探测每个点的四个方向
int fx[5] = {0,0,1,0,-1};
int fy[5] = {0,1,0,-1,0};
int tx,ty;//表示尝试访问的点
int k=1;//每个点标记的值, 每标记一次, 自增1
int m,n,i,j;
//head: 头指针, 指向头的位置, tail: 尾指针, 指向尾的位置
int head=1,tail=1;
int main(){
    cin>>n>>m;
    //起点
    a[1][1]=k;
    k++;
    //1,1点存入队列, 先以1,1作为head, 访问其四个方向
    q[1][1]=1;
    q[1][2]=1;
    //当head<=tail时, 说明队列还有点没有尝试标记过四个方向
    while(head<=tail){
        //尝试head对应的点的四个方向
        for(i=1;i<=4;i++){
            //从head对应的点开始, 加上方向值的变化, 得到四个新点
            tx=q[head][1]+fx[i];
            ty=q[head][2]+fy[i];
            //如果新点在矩阵内, 说明没有访问过
            if(tx>=1&&tx<=n&&ty>=1&&ty<=m&&a[tx][ty]==0){
                //新点存入队列
                tail++;
                q[tail][1]=tx;
                q[tail][2]=ty;
                a[tx][ty]=k;//标记访问的点对应的值
                k++;
            }
        }
        //head++使得刚才标记过的四个方向的点出队列, 取下一个点
    }
}

```

```

        head++;
    }
    //输出矩阵
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            cout<<a[i][j]<" ";
        }
        cout<<endl;
    }

    return 0;
}

```

## 2. 【提高】泉水

### 题目描述

Leyni是一个地址调查员，有一天在他调查的地方突然出现个泉眼。由于当地的地势不均匀，有高有低，他觉得如果这个泉眼不断的向外溶出水来，这意味着这里在不久的将来将会一个小湖。水往低处流，凡是比泉眼地势低或者等于的地方都会被水淹没，地势高的地方水不会越过。而且又因为泉水比较弱，当所有地势低的地方被淹没后，水位将不会上涨，一直定在跟泉眼一样的水位上。

由于Leyni已经调查过当地很久了，所以他手中有这里地势的详细数据。所有的地图都是一个矩形，并按照坐标系分成了一个个小方格，Leyni知道每个方格的具体高度。我们假定当水流到地图边界时，不会留出地图外，现在他想通过这些数据分析出，将来这里将会出现一个多大面积的湖。

### 输入

有若干组数据，每组数据的第一行有四个整数n,m,p1,p2(0<=1000)，n和m表示当前地图的长和宽，p1和p2表示当前地图的泉眼位置，即第p1行第p2列，随后的n行中，每行有m个数据。表示这每一个对应坐标的高度。

### 输出

输出对应地图中会有多少个格子被水充满。

### 样例输入

```

3 5 2 3
3 4 1 5 1
2 3 3 4 7
4 1 4 1 1

```

### 样例输出

```

6

```

### 思路：

模拟一下整个过程

### 示例代码：

```

#include<bits/stdc++.h>
using namespace std;
int n,m,p1,p2;
int a[1010][1010];
bool f[1010][1010];//标记某个点是否走过

```

```

int q[1000100][3];//队列数组
int fx[5]={0,0,1,0,-1};//方向数组
int fy[5]={0,1,0,-1,0};
int main(){
    int i,j,head=1,tail=1;
    int t,x,y;
    cin>>n>>m>>p1>>p2;
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            cin>>a[i][j];
        }
    }
    q[head][1]=p1;
    q[head][2]=p2;
    f[p1][p2]=true;
    while(head<=tail){
        for(i=1;i<5;i++){
            x=q[head][1]+fx[i];
            y=q[head][2]+fy[i];
            //在范围内且未访问过且小于等于泉眼高度
            if(x<=n&&x>=1&&y<=m&&y>=1&&a[x][y]<=a[p1][p2]&&f[x][y]==false){
                tail++;
                q[tail][1]=x;
                q[tail][2]=y;
                f[x][y]=true;//标记走过
            }
        }
        head++;
    }
    cout<<tail;
    return 0;
}

```

### 3. 【基础】走出迷宫的最少步数

#### 题目描述

一个迷宫由R行C列格子组成，有的格子里有障碍物，不能走；有的格子是空地，可以走。

给定一个迷宫，求从左上角走到右下角最少需要走多少步(数据保证一定能走到)。只能在水平方向或垂直方向走，不能斜着走。

#### 输入

第一行是两个整数，R和C，代表迷宫的行数和列数。（ $1 \leq R, C \leq 40$ ）

接下来是R行，每行C个字符，代表整个迷宫。空地格子用'.'表示，有障碍物的格子用'#'表示。迷宫左上角和右下角都是'.'。

#### 输出

输出从左上角走到右下角至少要经过多少步（即至少要经过多少个空地格子）。计算步数要包括起点和终点。

#### 样例输入

```
5 5
..####
#....#
#.#.#
#.#.#
#.#. .
```

### 样例输出

```
9
```

### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int n,m,tx,ty;
char a[50][50];
int q[2500][4];//队列数组
int fx[5]={0,0,1,0,-1};//方向数组
int fy[5]={0,1,0,-1,0};
int i,j,head=1,tail=1;
int main(){
    cin>>n>>m;
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            cin>>a[i][j];
        }
    }
    //1,1直接存入队列，从1,1,开始探索
    q[1][1]=1;
    q[1][2]=1;
    q[1][3]=1;//记录步数
    while(head<=tail){
        for(i=1;i<5;i++){
            tx=q[head][1]+fx[i];
            ty=q[head][2]+fy[i];
            if(a[tx][ty]=='.'){
                tail++;
                q[tail][1]=tx;
                q[tail][2]=ty;
                a[tx][ty]='#';
                q[tail][3]=q[head][3]+1;
                if(tx==n&&ty==m){
                    cout<<q[tail][3];
                    return 0;
                }
            }
        }
        head++;
    }
    return 0;
}
```

### 注意：

本题不可以将 $q[tail][3]$ 作为走到右下角的最小步数，因为右下角不一定是最后一个进入队列的点。

比如：

	j=1	j=2	j=3
i=1	.	#	.
i=2	.	#	.
i=3	.	.	.

最后一个进入队列的点是1,3

## 4. 【提高】走出迷宫的最短路径

### 1. 题目描述

有 $n*m$ 的迷宫，该迷宫有一个入口，一个出口。编写一程序打印一条从迷宫入口到出口的最短路径，黑色方块的单元表示走不通（用1表示），白色方块的内容表示走的通（用0表示）只能往上下左右四个方向走，如果有最短路径，保证最短路径一定是唯一的，如果没有路径可以到达，则输出“no way”。

### 输入

第一行输入2个整数n和m(n和m都是10~150之间的整数)，代表迷宫的行数和列数

接下来n行，每行有m个整数，1代表不可走的点，0代表可走的点

接下来一行，有2个整数s1和s2代表入口的坐标

接下来一行，有2个整数e1和e2代表出口的坐标

本题数据上保证出发点和终点的值一定为0，也就是不存在出发点和终点不能走的情况

### 输出

输出从入口到出口的最短路径，如果没有路径可达输出“no way”

### 样例输入

```
8 5
1 1 1 1 1
0 0 0 0 1
1 1 1 0 1
1 0 0 0 1
1 0 0 1 1
1 0 0 0 1
1 1 1 0 1
1 0 0 0 1
2 1
8 4
```

### 样例输出

```
(2,1)->(2,2)->(2,3)->(2,4)->(3,4)->(4,4)->(4,3)->(5,3)->(6,3)->(6,4)->
(7,4)->(8,4)
```

### 思路

记录遍历过程中的每个点，以及每个点的来源下标，到终点时递归寻找每个点的来源并打印即可

### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
int n,m,i,j,tx,ty;
int a[150][150];
int q[40000][4];
int fx[5]={0,0,1,0,-1};
int fy[5]={0,1,0,-1,0};
int head = 1,tail = 1;
int s1,s2,e1,e2;//开始、结束坐标
void print(int k){
    //如果前驱结点不为0
    if(q[k-1][3]!=0){
        print(q[k][3]);
    }
    cout<<"("<<q[k][3]<<","<<q[k][2]<<")";
    if(k!=tail){
        cout<<"->";
    }
}

int main(){
    cin>>n>>m;
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            cin>>a[i][j];
        }
    }
    cin>>s1>>s2>>e1>>e2;
    q[1][1]=s1;
    q[1][2]=s2;
    q[1][3]=0;
    while(head<=tail){
        for(i=1;i<5;i++){
            tx=q[head][1]+fx[i];
            ty=q[head][2]+fy[i];
            if(tx>=1&&tx<=n&&ty>=1&&ty<=m&&a[tx][ty]==0){
                tail++;
                q[tail][1]=tx;
                q[tail][2]=ty;
                a[tx][ty]=1;
                q[tail][3]=head;
                if(tx==e1&&ty==e2){
                    print(tail);
                    return 0;
                }
            }
        }
        head++;
    }
    cout<<"no way";
    return 0;
}
```

```
}
```

## 5. 【提高】骑士牛

### 题目描述

John用他的一头母牛和Don先生交换了一头“骑士牛”。这头牛有一个独特的能力——在牧场中能像中国象棋中的马一样跑跳（会中国象棋吗？不会？注意：本题不考虑马被“蹩脚”的情况）。当然，这头牛不能跳到岩石或树上，不过能跳到有牧草的地方。这儿有一个宽为X，高为Y的矩形牧场( $1 \leq X \leq 150; 1 \leq Y \leq 150$ )。“骑士牛”和其它牛一样喜欢干草。给你一张包含“骑士牛”出发地和树、岩石、灌木或其它障碍物及大包干草等位置信息的地图，确定“骑士牛”得到干草最少要跳几“跳”。地图中“骑士牛”出发地用‘K’表示；障碍物用‘\*’表示，牧草用‘.’表示，干草所在地用‘H’表示。这儿有一个示例地图：

	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8	j=9	j=10
i=1	.	.	.	.	.	.	.	.	.	.
i=2	.	.	.	.	*	.	.	.	.	.
i=3	.	.	.	.	.	.	.	.	.	.
i=4	.	.	.	*	.	*	.	.	.	.
i=5	.	.	.	.	.	.	.	*	.	.
i=6	.	.	*	.	.	*	.	.	.	H
i=7	*	.	.	.	.	.	.	.	.	.
i=8	.	.	*	.	.	.	*	.	.	.
i=9	.	K	.	.	.	.	.	.	.	.
i=10.	.	.	*	.	.	.	.	.	*	.
i=11.	.	*	.	.	.	.	*	.	.	.

骑士牛得到干草的最少步骤在下图中用ABC.....表示，最少要跳5“跳”（其它的路径可能超过5“跳”）：

	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8	j=9	j=10
i=1	.	.	.	.	.	.	.	.	.	.
i=2	.	.	.	*	.	.	.	.	.	.
i=3	.	.	.	.	.	.	.	.	.	.
i=4	.	.	*	.	*	.	.	.	.	.
i=5	.	.	.	.	.	.	*	.	.	.
i=6	.	*	.	.	*	.	.	.	.	H
i=7	*	.	B	.	.	.	.	.	.	.
i=8	.	.	*	C	.	.	*	E	.	.
i=9	K	A	.	.	.	D	.	.	.	.
i=10.	.	*	.	.	.	.	.	.	*	.
i=11.	.	*	.	.	.	.	*	.	.	.

输入

第1行: 两个空格隔开的整数: X 和 Y

第2..Y+1行: 第Y-i+2行包含X个没有空格的字符 (就像上面的地图一样) : 表示第i行的地图。

### 输出

第1行: 一个单独的整数表示最少的得到干草的“跳”数。所有的数据都能得到干草。

### 样例输入

```
10 11
.....
....*.....
.....
...*.*....
.....*..
..*..*...H
*.....
...*...*..
.K.....
...*....*
..*....*..
```

### 样例输出

```
5
```

## 2.课后作业

### 1. 【基础】迷宫出口

#### 题目描述

一天Extense在森林里探险的时候不小心走入了一个迷宫，迷宫可以看成是由 $n * n$ 的格点组成，每个格点只有2种状态，0和1，前者表示可以通行后者表示不能通行。同时当Extense处在某个格点时，他只能移动到东南西北(或者说上下左右)四个方向之一的相邻格点上，Extense想要从点A走到点B，问在不走出迷宫的情况下能不能办到。如果起点或者终点有一个不能通行(为1)，则看成无法办到。

#### 输入

第1行是一个正整数n ( $1 \leq n \leq 100$ )，表示迷宫的规模是 $n * n$ 的。接下来是一个 $n * n$ 的矩阵，矩阵中的元素为0或者1。再接下来一行是4个整数ha la hb lb，描述A处在第ha行 第la列，B处在第hb行 第lb列。

#### 输出

能办到则输出“YES”，否则输出“NO”。

### 样例输入

```
3
0 1 1
0 0 1
1 0 0
1 1 3 3
```

### 样例输出

YES

## 2. 【基础】数池塘（四方向）

### 题目描述

农夫约翰的农场可以表示成 $N \times M$  ( $1 \leq N \leq 100 \leq M \leq 100$ ) 个方格组成的矩形。由于近日的降雨，在约翰农场上的不同地方形成了池塘。每一个方格或者有积水 ('W') 或者没有积水 ('.')。农夫约翰打算数出他的农场上共形成了多少池塘。一个池塘是一系列相连的有积水的方格，每一个方格周围的四个方格都被认为是与这个方格相连的。现给出约翰农场的图样，要求输出农场上的池塘数。

### 输入

第1行：由空格隔开的两个整数：N和M

第2..N+1行：每行M个字符代表约翰农场的一排方格的状态。每个字符或者是'W'或者是'.'，字符之间没有空格。

### 输出

输出只有1行，输出约翰农场上的池塘数

### 样例输入

```
10 12
W.....WW.
.WWW.....WWW
....WW...WW.
.....WW..
.....W..
..W.....W..
.W.W.....WW.
W.W.W.....W.
.W.W.....W.
..W.....W.
```

### 样例输出

```
13
```

## 3. 【基础】走出迷宫的最少步数2

### 题目描述

当你站在一个迷宫里的时候，往往会被错综复杂的道路弄得失去方向感，如果你能得到迷宫地图，事情就会变得非常简单。假设你已经得到了一个 $n \times m$ 的迷宫的图纸，请你找出从起点到出口的最短路。

### 输入

第一行是两个整数n和m( $1 \leq n, m \leq 100$ )，表示迷宫的行数和列数。接下来n行，每行一个长为m的字符串，表示整个迷宫的布局。字符'.'表示空地，'#'表示墙，'S'表示起点'T'表示出口。

### 输出

输出从起点到出口最少需要走的步数

### 样例输入

```
3 3
```

```
S#T
```

```
.#. .
```

```
...
```

## 样例输出

```
6
```

### 4. 【提高】小X学游泳(swim)

#### 题目描述

暑假快到啦，小X准备趁着这个暑假去学游泳。可是一开始小X就遇到了一个难题。

游泳池划分成了一个  $n \times m \times m \times m$  的方格，这里  $n \times m \times m \times m$  表示  $n$  行  $m$  列。因为游泳池里的水深浅不一，所以这  $n \times m \times m \times m$  个方格对于小X的危险系数也会不一样。

而小X目前需要从左上角的方格  $(1, 1)$  出发，游到右下角的方格  $(n, m, m)$ ，小X每次只能从当前方格游到上下左右四个相邻的方格中的某一格，并且在到达终点前不能离开游泳池。

小X很担心会发生什么危险，所以希望你能帮他找一条危险系数最小的路径。

一条路径的危险系数定义为这条路径所经过的方格的危险系数之和。

注意：这条路径不能经过同一个方格两次（小X当然不希望去那么危险的地方再游一次）

#### 输入

输入数据第一行有两个用空格隔开的正整数  $n$  和  $m$ ，表示泳池的行数和列数。

接下来共有  $n$  行数据，每行有  $m$  个用空格隔开的大于等于 0 的整数，表示每个方格的危险系数

#### 输出

输出仅有一行包含一个整数  $ans$ ，表示要求的从左上角的方格  $(1, 1)$  出发，游到右下角的方格  $(n, m, m)$  的最小的危险系数。

#### 样例输入

```
4 5
1 7 2 8 2
3 10 1 5 1
2 8 3 7 1
1 2 1 20 1
```

## 样例输出

```
19
```

### 5. 【提高】小X学游泳

#### 题目描述

小X想要学游泳。

这天，小X来到了游泳池，发现游泳池可以用  $N$  行  $M$  列的格子来表示，每个格子的面积都是 1，且格子内水深相同。

由于小X刚刚入门，他只能在水深相同的地方游泳。为此，他把整个游泳池看成若干片区域，如果两个格子相邻（上下左右四个方向）且水深相同，他就认为它们属于同一片区域。

小X想知道最大的一片区域面积是多少，希望你帮帮他。

#### 输入

第一行包含用一个空格隔开的两个整数N,M。 (1≤N,M≤100)  
接下来N行，每行包含M个 1到9的数字，表示每个格子的水深

### 输出

第一行包含一个整数，表示最大的一片区域面积。

### 样例输入

```
3 3  
124  
224  
152
```

### 样例输出

```
3
```

### 说明

#### 数据范围

对于30%的数据， $1 \leq N, M \leq 3$ 。

对于60%的数据， $1 \leq N, M \leq 10$ 。

对于 100% 的数据， $1 \leq N, M \leq 100$ 。

#### 建议

请使用深搜和广搜分别实现

## 6. 【提高】方格取数

### 题目描述

在  $n$  行、 $m$  列的方格矩阵中，每个方格都包含一个数字。小明可以从任意方格出发开始移动。每次移动可以移到与当前方格有一条边相邻的方格(即向上、下、左或右方向移动 1 格，且不能移出边界)。除此之外，你移动到的方格中的数字必须比当前方格中的数字更大。

请你帮助小明编程规划移动路径，使路径上经过的所有数字之和最大。

本题方格中的数据根据输入的初始数字  $s$  按照如下算法生成：

```
for i = 1, 2, ... n  
for j = 1, 2, ... m  
s ← (s × 345) mod 19997  
矩阵第 i 行第 j 列方格中的数字为(s mod 10) + 1
```

### 输入

正整数  $n, m$  (方格的大小),  $s$  (数据生成器的初始数值)。 $1 \leq n, m \leq 100$ ,  $1 \leq s \leq 19,997$ 。

### 输出

所有合法路径中的最大数字和。

### 样例输入

```
4 5 97
```

### 样例输出

```
24
```

### 样例输入

40 50 1

### 样例输出

47

### 说明

样例1输入4 5 97，就会产生如下图所示的矩阵，那么所有合法路径中最大数字和为： $4 + 5 + 7 + 8 = 24$ 。

9	7	10	10	8
2	9	2	5	3
2	5	5	7	7
5	8	4	8	5

## 7. 【提高】最小拐弯路径

### 题目描述

农夫约翰在农场工作了一天，感觉比较累，准备开车回家。约翰在比较累的时候，喜欢走直路，不喜欢拐弯，哪怕走少拐弯的路回家更远，约翰也想走直路（好任性的约翰!）。请你从约翰的出发地到目的地找一条路，使得约翰回家拐弯数量最少。

### 输入

第一行两个整数n和m（n和m都是1000以内的整数），代表地图的大小。

接下来的n行，每行有m个数，其中能够通行的点用0表示，不能通行的点用1表示。

再接下来1行，有4个整数，s1、s2、e1、e2，s1和s2表示出发点的坐标，e1和e2表示目的地的坐标。（本题测试数据保证起点和终点不是同一个点，且起点和终点一定可以走）

### 输出

约翰从出发点到目的地最少要拐弯的数量，本题所有数据都确认从出发点到目的地是有路径可达的。

### 样例输入

5	7					
1	0	0	0	0	1	0
0	0	1	0	1	0	0
0	0	0	0	1	0	1
0	1	1	0	0	0	0
0	0	0	0	1	1	0
1	3	1	7			

### 样例输出

5

# 第三十二章 指针与结构体

## 1. 什么是指针？

### 1.1 概念

指针 (Pointer) : 变量的地址。通过它可以找到以它为地址的内存单元。

### 1.2 入门案例

#### 1.2.1 理解指针的概念

例子：理解指针的概念，区分什么是地址，什么是地址指向的值

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    //整数变量
    int x=10;
    //定义指针（整数指针），指向x的地址
    //p是int*类型（整数指针）
    //&表示取变量的地址
    int *p = &x;
    cout<<p<<endl;
    //p是地址， *p不是， *p表示p指向的地址对应的值
    cout<<*p<<endl;

    //通过指针，修改变量的值
    //p指向的值，自增2
    *p=*p+2; //与p=p+2完全是两件事， p是地址， *p不是
    cout<<x<<endl;
    cout<<p<<" "<<*p;
    return 0;
}
```

例子：对比如下代码输出的结果，理解指针和普通变量的不同

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int x =10;
    //值拷贝，将x的值拷贝给y
    int y=x;
    y=y+2;
    cout<<x<<endl;
    //将x的地址拷贝给p
    int *p=&x;
    *p=*p+2;
    cout<<x<<endl;
    return 0;
}
```

**注意：**值拷贝与地址拷贝是不同的~

### 1.2.2 &与\*的嵌套使用

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int x=10;
    int *p=&x;
    cout<<p<<endl;
    cout<<*p<<endl;
    cout<<&(*p)<<endl;
    cout<<*&(*p)<<endl;
    return 0;
}
```

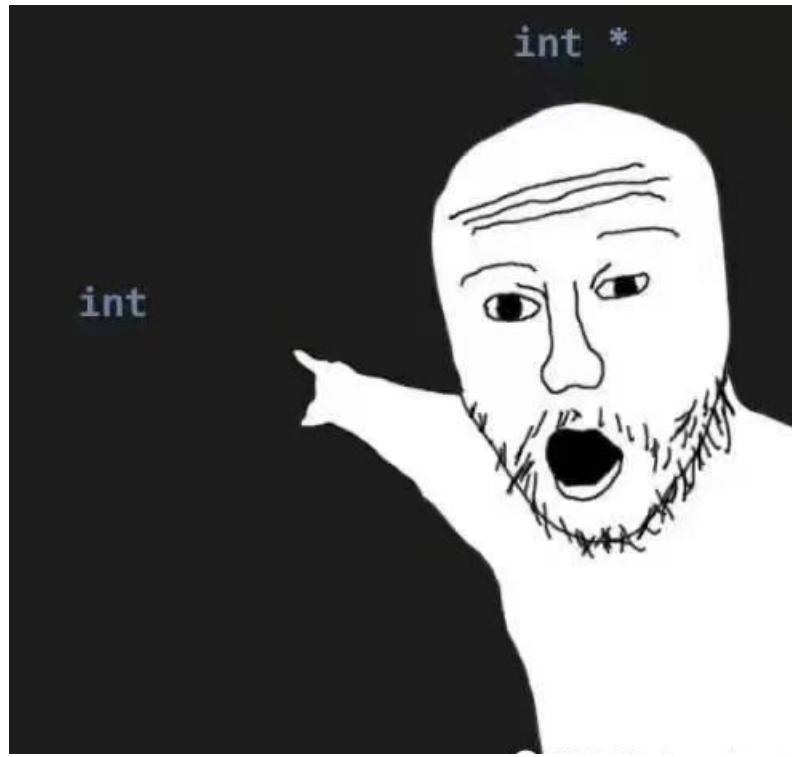
### 1.2.3 \*p++和(\*p)++的区别

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int x=10;
    int *p=&x;
    cout<<p<<" "<<*p<<endl;
//    *p++;
    (*p)++;
    cout<<p<<" "<<*p;
    return 0;
}
```

**注意：**++的运算优先级要高于\*，因此\*p++相当于p++（即地址自增），然后再取地址对应的值

## 2.指针的作用

### 2.1 通过函数借助指针修改变量的值



示例代码：

```
#include<bits/stdc++.h>
using namespace std;
void change1(int x){
    x++;
}
void change2(int *p){
    (*p)++;
}
int main(){
    int x=1;
    change1(x);
    cout<<x<<endl;
    int p=1;
    change2(&p);
    cout<<p;
    return 0;
}
```

image-20220921130422621

注意：不同的函数内部定义的变量，存储在不同的栈内存中

## 2.2 让函数返回多个值

```
#include<bits/stdc++.h>
using namespace std;
//通过函数计算两个数的较大、较小、平均值
double num(int a,int b,int *max,int *min){
    if(a>b){
        *max = a;
        *min = b;
    }else{
```

```

        *max = b;
        *min = a;
    }
    return (a+b)/2.0;
}
int main(){
    int a,b,max,min;
    cin>>a>>b;
    double avg = num(a,b,&max,&min);
    cout<<max<<" "<<min<<" "<<avg;
    return 0;
}

```

## 2.3 在scanf中使用指针

**补充:**

scanf() 和 printf() 都是C语言的写法，而C++兼容C语言的这种写法，反过来不行。

作用等同于cin 和cout

**使用语法:**

scan(数据格式, 地址)

printf(输出语句与占位符, 数据1, 数据2, 数据3...)

**格式说明:**

%d:整数 %f: float %lf: double %c: 字符 %p: 指针

**示例代码:**

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int a,b;
    scanf("%d%d",&a,&b);
    printf("%d+%d的值是: %d\n",a,b,a+b);
    int *p=&a;
    printf("%p 指向的值是: %d",p,*p);
    return 0;
}

```

## 3.数组指针

### 1.数组就是指针

**示例代码:**

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int a[5]={1,2,3,4,5};
    //数组的本质是指向a[0]的地址
    cout<<a<<" "<<&a<<" "<<&a[0]<<endl;
    //定义指针, 指向数组
}

```

```

int *p = a;
cout<<p<<" "<<a[0]<<endl;
*p=*p+2;
cout<<p<<" "<<a[0]<<endl;
//修改a[1]的值
p++;
*p=*p+5;
cout<<p<<" "<<a[1]<<endl;

return 0;
}

```

## 2.函数中，通过指针输出数组元素

```

#include<bits/stdc++.h>
using namespace std;
/*
向函数中传递整数数组，必须传递的数组的长度
因为本上传递过来的是a[0]的地址
*/
void fun1(int a[],int n){
    for(int i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    cout<<endl;
}
/*
向函数中传递数组，直接写成指针形式
*/
void fun2(int *a,int n){
    for(int i=0;i<n;i++){
        cout<<*a<<" ";
        a++;
    }
    cout<<endl;
}
/*
输出字符数组里的每个字符
*/
void fun3(char s[]){
    for(int i=0;i<strlen(s);i++){
        cout<<s[i]<<" ";
    }
    cout<<endl;
}
/*采用指针的方式遍历*/
void fun4(char *s){
    while(*s!='\0'){
        cout<<*s<<" ";
        s++;
    }
    cout<<endl;
}
int main(){
    int a[3]={5,4,3};

```

```
// fun1(a,3);
// fun2(a,3);
char s[10]={"coderbee"};
fun4(s);
return 0;
}
```

## 4.结构体

### 4.1 什么是结构体

结构体是用户自定义的允许存储不同类型数据项的数据结构。 (数组只能允许同一种数据类型)

**入门案例：**

如果需要存储一个学生的学号、姓名、身高、年龄、家庭住址信息，而这些变量都是不同的数据类型，不适合定义数组来存储，这时，就可以使用结构体来实现。

**示例代码**

```
#include<bits/stdc++.h>
using namespace std;
/*
结构体可自定义为任意类型，本案例中定义为Student类型
结构体命名规则：驼峰命名法（所有单词首字母大写）
*/
struct Student{
    //定义成员变量
    int num;//学号
    string name;//姓名
    double height;//身高
}s1,s2;//可以定义结构体的同时定义结构体变量，也可以不定义
//定义输出结构体信息的函数
void print(Student s){
    cout<<"num:"<<s.num<<" name:"<<s.name<<" height:"<<s.height<<endl;
}
int main(){
    //定义结构体Student类型的变量
    struct Student s;
    //为结构体s的属性（成员变量）赋值
    s.num=14;
    s.name="小兰";
    s.height=153.5;
    print(s);
    s.height=132;
    s.name="小绿";
    s.num=5;
    print(s);
    return 0;
}
```

**注意：**

1. 可以在定义结构体的同时定义结构体变量，也可以不同时定义
2. 定义结构体不代表定义结构体变量

### 3. 引用结构体成员变量的方法是：结构体变量.成员变量名

#### 结构体数组的定义与使用

```
#include<bits/stdc++.h>
using namespace std;
/*
结构体可自定义为任意类型，本案例中定义为Student类型
结构体命名规则：驼峰命名法（所有单词首字母大写）
*/
struct Student{
    //定义成员变量
    int num;//学号
    string name;//姓名
    double height;//身高
}s1,s2;//可以定义结构体的同时定义结构体变量，也可以不定义
//定义输出结构体信息的函数
void print(Student s[],int n){
    for(int i=0;i<n;i++){
        cout<<"num:"<<s[i].num<<" name:"<<s[i].name<<" height:"
        <<s[i].height<<endl;
    }
}
int main(){
    //定义结构体Student类型的数组
    Student s[10];
    int n;
    cin>>n;
    //为结构体s的属性（成员变量）赋值
    for(int i=0;i<n;i++){
        cin>>s[i].num>>s[i].name>>s[i].height;
    }
    print(s,n);
    return 0;
}
```

#### 结构体指针的使用

```
#include<bits/stdc++.h>
using namespace std;
/*
结构体可自定义为任意类型，本案例中定义为Student类型
结构体命名规则：驼峰命名法（所有单词首字母大写）
*/
struct Student{
    //定义成员变量
    int num;//学号
    string name;//姓名
    double height;//身高
};
int main(){
    Student s;
    s.num=14;
    s.name="小兰";
}
```

```

s.height=153.5;
cout<<"num:"<<s.num<<" name:"<<s.name<<" height:"<<s.height<<endl;

//定义结构体指针
Student *p=&s;
//定义结构体指针后使用成员变量需要用 指针->成员名 的形式
cout<<p->num<<" "<<p->name<<" "<<p->height<<endl;
//p是指针, *p是结构体
cout<<(*p).num<<" "<<(*p).name<<" "<<(*p).height<<endl;

return 0;
}

```

注意: ->表示 指向

## 4.2 结构体的应用

### 4.2.1 课堂练习

#### 1. 【基础】期末考试成绩排名

##### 题目描述

期末考试结束了，数学成绩已经出来，数学老师请你帮忙编写一个程序，可以帮助老师对班级所有同学的考试分数按照由高到低进行排序，并输出按照成绩排序后每个同学的学号、姓名、数学成绩。

##### 输出

第一行是一个整数n (n<=100) , 代表班级的总人数

接下来n行, 每行有3个数据, 第一个数据是某个同学的学号, 第二个数据是该同学的姓名的拼音(拼音不含空格), 第三个数据是该同学的数学成绩 (成绩是整数)

##### 输出

按照数学成绩由高到低输出每个同学的学号、姓名、数学成绩, 每行含1个同学的3个数据, 3个数据用空格隔开。 (如果出现多个同学数学成绩相同, 则按照学号由小到大输出, 不存在多个同学学号相同的情况)

##### 样例输入

```

3
1 zhangfang 98
2 liming 100
3 sunhua 99

```

##### 样例输出

```

2 liming 100
3 sunhua 99
1 zhangfang 98

```

##### 解法一：使用结构体数组，冒泡排序

```

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

```

```

struct Student{
    int num;
    string name;
    int score;
};

int main(){
    Student a[110];
    int n,i,j;
    cin>>n;
    for(int i=0;i<n;i++){
        cin>>a[i].num>>a[i].name>>a[i].score;
    }
    for(i=0;i<n-1;i++){
        for(j=0;j<n-i-1;j++){
            if(a[j].score<a[j+1].score ||
                (a[j].score==a[j+1].score&&a[j].num<a[j+1].num)){
                swap(a[j],a[j+1]);
            }
        }
    }
    for(i=0;i<n;i++){
        cout<<a[i].num<<" "<<a[i].name<<" "<<a[i].score<<endl;
    }
    return 0;
}

```

## 解法二：使用sort排序

```

#include<bits/stdc++.h>
using namespace std;
struct Student{
    int num;
    string name;
    int score;
};

//sort的比较函数
bool cmp(Student s1,Student s2){
    //希望满足该规则排序, return true
    if(s1.score>s2.score || (s1.score==s2.score&&s1.num<s2.num)){
        return true;
    }else{
        return false;
    }
}

int main(){
    Student a[110];
    int n,i,j;
    cin>>n;
    for(int i=0;i<n;i++){
        cin>>a[i].num>>a[i].name>>a[i].score;
    }
    sort(a,a+n,cmp);
    for(i=0;i<n;i++){
        cout<<a[i].num<<" "<<a[i].name<<" "<<a[i].score<<endl;
    }
}

```

```
    return 0;  
}
```

## 2. 【入门】遥控飞机争夺赛

### 题目描述

红太阳杯遥控飞机大赛拉开帷幕。比赛规则为，每位选手让自己的飞机从起点到终点飞行5次，组委会记录5次的飞行的成绩之后去掉一个最大成绩、一个最小成绩后计算剩余3个成绩的平均分（平均分保留3位小数）作为该选手的最终成绩。

有n名选手参加了比赛，从键盘读入每位选手的编号以及他们的5次飞行的成绩。

请根据n名选手的比赛成绩，编程计算出冠军、亚军、季军的编号以及组委会计算出的成绩。（假设不存在多名选手成绩正好一样）（4.1.51）

### 输出

第一行为一个整数n，代表参加比赛的选手数量 ( $n \geq 4 \&& n \leq 100$ )

后面的n行，每行有6个数，第一个数是选手的编号，后5个数为选手的5次飞行的成绩

### 输出

3行，第一行输出冠军的编号及飞行成绩（保留3位小数）用空格隔开2个数；第二行输出亚军的编号及飞行成绩，第三行输出季军的编号及飞行成绩

### 样例输入

```
4  
11 58 59 60 61 62  
18 59 60 61 62 63  
23 65 64 63 62 62  
10 60 61 61 65 62
```

### 样例输出

```
23 63.000  
10 61.333  
18 61.000
```

### 示例代码

```
#include<bits/stdc++.h>  
using namespace std;  
struct Player{  
    int num;  
    double score;  
};  
bool cmp(Player p1,Player p2){  
    if(p1.score>p2.score){  
        return true;  
    }else{  
        return false;  
    }  
}  
int main(){
```

```

Player a[110];
int n,i,j,ma,mi,s,x;
cin>>n;
for(i=0;i<n;i++){
    cin>>a[i].num;
    s=0;
    ma=INT_MIN;
    mi=INT_MAX;
    for(j=0;j<5;j++){
        cin>>x;
        s+=x;
        ma = max(ma,x);
        mi = min(mi,x);
    }
    a[i].score=(s-ma-mi)/3.0;
}
sort(a,a+n,cmp);
for(i=0;i<3;i++){
    cout<<a[i].num<<" "<<fixed<<setprecision(3)<<a[i].score<<endl;
}
return 0;
}

```

### 3. 【基础】活动选择

#### 题目描述

学校在最近几天有  $n$  ( $n \leq 100$ ) 个活动，这些活动都需要使用学校的大礼堂，在同一时间，礼堂只能被一个活动使。由于有些活动时间上有冲突，学校办公室人员只好让一些活动放弃使用礼堂而使用其他教室。

现在给出  $n$  个活动使用礼堂的起始时间  $begin\_i$  和结束时间  $end\_i$ 。 $(begin\_i < end\_i)$ ，请你帮助办公室人员安排一些活动来使用礼堂，要求安排的活动尽量多。请问最多可以安排多少活动？

请注意，开始时间和结束时间均指的是某个小时的 000 分 000 秒，如：333 555，指的是 3:003:003:00~5:005:005:00，因此333 555和555 999这两个时间段不算冲突的时间段。

#### 输出

第一行一个整数  $n$  ( $n \leq 100$ )；

接下来的  $n$  行，每行两个整数，第一个  $begin\_i$ ，第二个是  $end\_i$  ( $begin\_i < end\_i \leq 32767$ )；

#### 输出

输出最多能安排的活动数；

#### 样例输入

```
11
3 5
1 4
12 14
8 12
0 6
8 11
6 10
5 7
3 8
5 9
2 13
```

### 样例输出

```
4
```

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
//Node:结点
//定义结构体存储活动的开始、结束时间
struct Node{
    int begin;
    int end;
}a[110];
//按照结束时间排序
bool cmp(Node n1,Node n2){
    if(n1.end<n2.end){
        return true;
    }else{
        return false;
    }
}
int main(){
    int i,n,c=0,e;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i].begin>>a[i].end;
    }
    sort(a,a+n,cmp);
    c=1;
    e=a[0].end;
    for(i=1;i<n;i++){
        if(a[i].begin>=e){
            c++;
            e=a[i].end;
        }
    }
    cout<<c;
    return 0;
}
```

## 4. 【入门】宇宙总统2

### 题目描述

地球历公元6036年，全宇宙准备竞选一个最贤能的人当总统，共有n个非凡拔尖的人竞选总统，现在投票已经结束，获得选票最多的人将荣登总统的宝座，如果有多个候选人获得票数一致，那么名字字典码最大的人将获得总统的宝座。请你编程计算出谁能够胜任总统的职位。

比如，有10个人参加了投票，这10张选票结果分别是。

```
zhangguoqiang  
liming  
wangfang  
zhangguoqiang  
wangfang  
zhangguoqiang  
zhaofei  
zhaofei  
wangfang  
zhaofei
```

统计结果zhangguoqiang、wangfang、zhaofei三人每人都3票，由于zhaofei名字的字典码最大，zhaofei当选为本届宇宙总统。

### 输出

第1行是一个整数n，代表投票的总数。（ $n \leq 1000$ ）

第2行~第n+1行，每行是一个得到选票的人的名字（名字一定是小写的拼音，没有空格）。

### 输出

输出n行，按照每个人的得票数由高到低的顺序输出每个人的名字和得票数，中间用空格隔开。

（如果有多人得票数一致，则名字字典码大的人排在前面）

### 样例输入

```
10  
liming  
wangfang  
zhangguoqiang  
zhangguoqiang  
wangfang  
zhangguoqiang  
zhaofei  
zhaofei  
wangfang  
zhaofei
```

### 样例输出

```
zhaofei 3  
zhangguoqiang 3  
wangfang 3  
liming 1
```

## 4.2.2 课后作业

### 1. 【入门】购买贺年卡

#### 题目描述

新年快到了，笑笑打算给他的好朋友们发贺年卡，而且他已经选好了自己要购买的贺卡的样式。俗话说得好，货比三家，笑笑来到商店，看了各个商铺这种贺卡的价钱。不仅如此，笑笑还记住了每个商铺的存货量。已知笑笑打算购买m张贺卡，问他最少花多少钱。

#### 输出

第一行两个整数m和n。其中m表示要购买的贺卡的数量，n表示商铺的个数。

以下n行，每行两个整数，分别表示该商铺这种贺卡的单价和存货量。

#### 输出

仅一个数，表示笑笑花的最少钱数。

#### 样例输入

```
10 4
4 3
6 2
8 10
3 6
```

#### 样例输出

```
36
```

#### 说明

数据规模： $0 < m, n \leq 1000$ ，每个商铺贺卡单价在1~100之间，数量在1~1000之间，输入保证商铺的总存货量不少于m。

### 2. 【基础】统计每个数出现的次数

#### 题目描述

从键盘读入n个数 ( $n \leq 1000$ )，统计每个数出现的次数，从小到大输出每个出现过的数，及每个数出现的次数。

比如：假设从键盘读入6个数，分别是：1 6 8 1 2 6，那么输出如下：

```
1 2
2 1
6 2
8 1
```

输出含义为：1出现了2次，2出现了1次，6出现了2次，8出现了1次。

#### 输出

第1行有一个整数n ( $n \leq 1000$ )

第2行有n个整数，数字之间用空格隔开，这n个数都是int范围内的数。

#### 输出

输出若干行，每行2个数，第1个数是出现过的数，第2个数是该数出现的次数，要求从小到大输出每个数及每个数出现的次数。

### 样例输入

```
10
2 8 1 2 3 3 6 1 1 1000
```

### 样例输出

```
1 3
2 2
3 2
6 1
8 1
1000 1
```

**提示：**用结构体存储每个数以及该数出现的次数

## 3. 【入门】等比例缩放照片

### 题目描述

如图所示的一张照片（图①），可以把它的宽度或者高度减小从而减少照片的尺寸（如图②、③、④）。但只有等比例缩放的情况下照片才是最好看的，如图④；图②照片被压扁，图③照片被拉长。

给定图片的原始尺寸以及n组要压缩的尺寸，请问哪组压缩后的数据的宽高比最接近原始数据？如果有多个压缩尺寸的宽高比都是一样的且都是最接近原始数据的，那么输出压缩后面积最小的那组数据。



### 输出

第1行，2个整数x和y，代表图片的原始尺寸的宽和高

第2行，一个整数n，代表接下来有n组压缩后的尺寸（ $n \leq 100$ ）

接下来n行，每行2个数，代表n组压缩后的宽和高（确保输入的宽  $\geq$  高）（本题所有照片的宽高均在1~10000之间）

### 输出

宽高比和原始图片最接近的宽高数据，如果有多个这样的数，输出面积最小的那组（不存在多组宽高比和原始数据一样接近且面积又一样的数据）

### 样例输入

```
10 4
4
20 4
60 10
15 9
10 6
```

### 样例输出

#### 4. 【基础】游览动物园

##### 题目描述

动物园有很多游览区，小红已经在动物园的一个游览区游览，突然接到电话，要半个小时内到动物园外面跟一个朋友见面。半个小时小红只够游览完当前区域之后，游览一个最近的景区。已知从一个游览区域只能沿着地图（地图的长宽均小于100）中的直线走（上下左右四个方向），请问离小红当前游览区的最近的一个游览区的坐标是多少，如果有多个点离小红的位置都很近，请输出离出口最近的那个点的坐标（不存在多个点距离出口一样近）？

例如：假设小红在孔雀区（3,2），离小红最近的2处游览区分别是猴山（2,0）和虎山（5,3），但猴山离入口更近，因此输出猴山的坐标。



##### 输出

第一行2个变量，为小红所在的游览区的坐标

第二行一个整数n，为该动物园内游览区的数量 ( $n \leq 100$ )

接下来的n行，每行2个数，代表动物园的n个游览区的坐标（本题所有的坐标值都在1~1000的范围内）

##### 输出

离小红最近的游览区的坐标

##### 样例输入

```
3 2
5
2 0
5 3
3 2
5 5
3 8
```

##### 样例输出

```
2 0
```

#### 5. 【提高】买木头

##### 题目描述

有n个木材供应商，每个供货商有长度相同一定数量的木头。长木头可以锯短，但短木头不能接长。有一个客人要求m根长度相同的木头。要求计算出，此时供货商提供的木头满足客人要求的最长的长度是多少。

例如 $n=2, m=30$ ，两个供货商的木头为

12,10 第1个供货商的木头长度为12，共有10根；

5,10 第2个供货商的木头长度为5，共有10根。

计算的结果为5，即长度为12的木头一根可锯出两根长度为5的木头，多余的无用，长度为5的木头不动，此时可得到30根长度为5的木头。

## 输出

整数n,m,L1,S1 (1≤n≤10000,1≤m≤1000000,1≤L1≤10000,1≤S1≤100)

其中L1是第一个供货商木头的长, S1是第一个供货商木头数量。其他供货商木头的长度和数量Li和Si (i≥2) , 由下面的公式给出:

$$Li=((L(i-1) \times 37011 + 10193) \bmod 10000) + 1$$

$$Si=((S(i-1) \times 73011 + 24793) \bmod 100) + 1$$

## 输出

一个整数, 即满足要求的m根长度相同的木头的最大长度。

## 样例输入

```
10 10000 8 20
```

## 样例输出

```
201
```

**提示:** 用结构体存储每个木头供货商能够提供的木头的长度和数量

## 6. 【入门】求最大梯形的面积

### 题目描述

从键盘读入n(3<=n<=100)个梯形的上底、下底和高, 请问这n个梯形中, 最大面积的梯形的面积是多少? (梯形面积的求解公式为  $S = (a + b) * h / 2$ , 也就是( $a + b$ ) \*  $h / 2$ )  
(5.1.18)

## 输出

第1行为1个整数n, 接下来n行每行3个整数分别代表梯形的上底、下底和高。

## 输出

最大面积梯形的面积 (结果保留1位小数)

## 样例输入

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

## 样例输出

```
17.5
```

## 7. 【提高】花生采摘

### 题目描述

鲁宾逊先生有一只宠物猴，名叫多多。这天，他们两个正沿着乡间小路散步，突然发现路边的告示牌上贴着一张小小的纸条：“欢迎免费品尝我种的花生！——熊字”。

鲁宾逊先生和多多都很开心，因为花生正是他们的最爱。在告示牌背后，路边真的有一块花生田，花生植株整齐地排列成矩形网格（如图1）。有经验的多多一眼就能看出，每棵花生植株下的花生有多少。为了训练多多的算术，鲁宾逊先生说：“你先找出花生最多的植株，去采摘它的花生；然后再找出剩下的植株里花生最多的，去采摘它的花生；依此类推，不过你一定要在我限定的时间内回到路边。”

我们假定多多在每个单位时间内，可以做下列四件事情中的一件：

- \1) 从路边跳到最靠近路边（即第一行）的某棵花生植株；
- \2) 从一棵植株跳到前后左右与之相邻的另一棵植株；
- \3) 采摘一棵植株下的花生；
- \4) 从最靠近路边（即第一行）的某棵花生植株跳回路边。

现在给定一块花生田的大小和花生的分布，请问在限定时间内，多多最多可以采到多少个花生？注意可能只有部分植株下面长有花生，假设这些植株下的花生个数各不相同。

例如在图所示的花生田里，只有位于(2, 5), (3, 7), (4, 2), (5, 4)的植株下长有花生，个数分别为13, 7, 15, 9。沿着图示的路线，多多在21个单位时间内，最多可以采到37个花生。



## 输出

输入数据的第一行包括三个整数，M, N和K，用空格隔开；表示花生田的大小为M \* N ( $1 \leq M, N \leq 20$ )，多多采花生的限定时间为K ( $0 \leq K \leq 1000$ ) 个单位时间。接下来的M行，每行包括N个非负整数，也用空格隔开；第*i* + 1行的第*j*个整数P<sub>ij</sub> ( $0 \leq P_{ij} \leq 500$ ) 表示花生田里植株(i, j)下花生的数目，0表示该植株下没有花生。

## 输出

输出包括一行，这一行只包含一个整数，即在限定时间内，多多最多可以采到花生的个数。

## 样例输入

```
6 7 21
0 0 0 0 0 0 0
0 0 0 0 13 0 0
0 0 0 0 0 0 7
0 15 0 0 0 0 0
0 0 0 9 0 0 0
0 0 0 0 0 0 0
```

## 样例输出

```
37
```

## 样例输入

```
6 7 20
0 0 0 0 0 0 0
0 0 0 0 13 0 0
0 0 0 0 0 0 7
0 15 0 0 0 0 0
0 0 0 9 0 0 0
0 0 0 0 0 0 0
```

## 第三十三章 STL (标准模板库)

### 1. STL的概念

STL简介：STL(standard template library)标准模板库，是“容器”的集合。

STL中常见的集合有：向量（vector）、栈（stack）、队列（queue）、优先队列（priority\_queue）、链表（list）、集合（set）、映射（map）等容器。

C++标准模板库的核心包括以下三个组件：

组件	描述
容器（Containers）	容器是用来管理某一类对象的集合。C++ 提供了各种不同类型的容器，比如 <code>deque</code> 、 <code>list</code> 、 <code>vector</code> 、 <code>map</code> 等。
算法（Algorithms）	算法作用于容器。它们提供了执行各种操作的方式，包括对容器内容执行初始化、排序、搜索和转换等操作。
迭代器（Iterators）	迭代器用于遍历对象集合的元素。这些集合可能是容器，也可能是容器的子集。

**补充：**

1. 容器：用来存储各种数据类型的数据结构（计算机存储、组织数据的方式），如数组
2. 迭代器：可依次存取容器中元素的东西，如数组 `int a[100]` 是个容器，`int *` 类型的指针变量就可以作为迭代器，用来遍历容器的所有元素
3. 算法：系统定义好的作用于容器的各种函数，如 `sort ()` 可以对一个 `vector` 中的数据进行排序，`find ()` 用来搜索一个 `list` 中的对象。

STL内所有组件都由模板（template）构成，其元素可以是任意类型。

**常用容器的分类：**

1. **常用的顺序容器：** `vector`, `deque`, `list`
2. **常用的管理容器：** `set`, `map`
3. **容器适配器：** `stack`, `queue`, `priority_queue`

**常用的容器算法：**

搜寻，排序，拷贝等

### 2. 向量（vector）

#### 2.1 什么是vector

向量（vector）是一个顺序容器（sequence container），它能够存放各种类型的对象，可以简单地认为，向量是一个能够存放任意类型的动态数组（元素个数可变）

## 2.2 vector的常见函数

函数名	函数说明
push_back(元素)	增加一个元素到向量后面
insert(位置, 元素)	插入元素到向量的指定位置
insert(位置, 个数n, 元素)	插入n个相同的元素到向量的指定位置
insert(位置, 向量头指针first, 向量尾指针end)	将另一个向量从first位置开始到end结束(不包括end)之间的内容插入当前向量的指定位置
erase (位置)	删除指定位置的元素
erase (开始位置, 结束位置)	删除向量中[first, last) 中元素
pop_back ()	弹出(删除)向量的最后一个元素
clear ()	清除向量所有元素, size()变成0
运算符[i]	取向量下标为i的元素
front ()	取向量的第一个元素
back ()	取向量的最后一个元素
begin ()	返回向量头指针(迭代器), 指向第一个元素
end ()	返回向量尾指针, 指向向量最后一个元素的下一个位置
rbegin ()	反向迭代器, 指向最后一个元素
rend ()	反向迭代器, 指向第一个元素之前的位置
size ()	返回向量中实际元素的个数
resize (大小)	重新设定向量的大小, 也就是可以保存元素的个数
max_size()	得到vector最大可以是多大
empty ()	判断向量是否为空, 等价于size()为0
swap ()	交换两个相同类型向量的数据

### 2.2.1 vector的存储与遍历

构造vector的四种常见方法:

1. vector(): 创建一个空的vector
2. vector(n): 创建一个元素个数为n的vector
3. vector(n,t): 创建一个元素为n且值为t的vector
4. vectoe(begin,end): 复制[begin,end) 区间内另一个数组的元素到vector中

### 方式一常规方法遍历

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

```

int main(){
    vector<int> v;//<>内可以放任意数据类型
    //size()用来获取vector实际存储的元素个数
    cout<<v.size()<<endl;
    /*
    v[0] = 5;
    v[1] = 8;
    //上面为常见错误，因为 vector<int> v 定义的是一个空的vector
    //不可以用下标来存储、访问元素，放入元素需要使用函数 push_back()
    */
    v.push_back(10);
    v.push_back(20);
    v.push_back(30);
    cout<<v.size()<<endl;
    //遍历vector中的所有元素
    for(int i=0;i<v.size();i++){
        //vector中有元素时可以使用下标访问，但不可越界
        cout<<v[i]<<" ";
    }
    return 0;
}

```

## 方式一自定义函数方法遍历

```

#include<bits/stdc++.h>
using namespace std;
//定义一个打印函数
void fun(vector<int> v){
    //遍历vector中的所有元素
    for(int i=0;i<v.size();i++){
        //vector中有元素时可以使用下标访问，但不可越界
        cout<<v[i]<<" ";
    }
}
int main(){
    vector<int> v;//<>内可以放任意数据类型
    //size()用来获取vector实际存储的元素个数
    cout<<v.size()<<endl;
    /*
    v[0] = 5;
    v[1] = 8;
    //上面为常见错误，因为 vector<int> v 定义的是一个空的vector
    //不可以用下标来存储、访问元素，放入元素需要使用函数 push_back()
    */
    v.push_back(10);
    v.push_back(20);
    v.push_back(30);
    cout<<v.size()<<endl;
    //调用打印函数
    fun(v);
    return 0;
}

```

## 方式二有初始长度且初始值为0:

```
#include<bits/stdc++.h>
using namespace std;
//定义一个打印函数
void fun(vector<int> v){
    //遍历vector中的所有元素
    for(int i=0;i<v.size();i++){
        //vector中有元素时可以使用下标访问，但不可越界
        cout<<v[i]<<" ";
    }
}
int main(){
    //定义一个长度为10默认值为0的vector
    vector<int> v(10);
    fun(v);
    cout<<endl;
    //会在默认值后面追加，而非改变初始的值，初始值可以
    //通过下标访问
    v.push_back(10);
    fun(v);
    return 0;
}
```

## 方式三带自定义初始值:

```
#include<bits/stdc++.h>
using namespace std;
//定义一个打印函数
void fun(vector<int> v){
    //遍历vector中的所有元素
    for(int i=0;i<v.size();i++){
        //vector中有元素时可以使用下标访问，但不可越界
        cout<<v[i]<<" ";
    }
}
int main(){
    //定义一个长度为10默认值为0的vector
    vector<int> v(10,123);
    fun(v);
    cout<<endl;
    //会在默认值后面追加，而非改变初始的值，初始值可以
    //通过下标访问
    v.push_back(10);
    fun(v);
    return 0;
}
```

## 方式四用数组初始化vector

```
#include<bits/stdc++.h>
```

```

using namespace std;
//定义一个打印函数
void fun(vector<int> v){
    //遍历vector中的所有元素
    for(int i=0;i<v.size();i++){
        //vector中有元素时可以使用下标访问，但不可越界
        cout<<v[i]<<" ";
    }
}
int main(){
    int a[]={1,2,3,4,5,6};
    //使用数组来初始化vector
    vector<int> v(a,a+6);
    //注意：这里也可以使用a+sizeof(a)/sizeof(a[0])来实现 [除以sizeof(int)也可以]
    //sizeof():计算变量占用的字节数，1字节=8位（1byte=8bit）
    fun(v);
    return 0;
}

```

## 2.2.2 vector的插入、删除、获取头尾元素

```

#include<bits/stdc++.h>
using namespace std;
//定义一个打印函数
void fun(vector<int> v){
    //遍历vector中的所有元素
    for(int i=0;i<v.size();i++){
        //vector中有元素时可以使用下标访问，但不可越界
        cout<<v[i]<<" ";
    }
}
int main(){
    vector<int> v;
    v.push_back(10);
    v.push_back(20);
    v.push_back(30);
    // cout<<"第一个元素："<<v.front()<<endl;
    // cout<<"最后一个元素："<<v.back()<<endl;
    //1.insert(位置, 元素):位置参数必须提供位置指针
    //v.insert(1,100) <==这种写法是错误的
    // v.insert(v.begin(),100);
    // fun(v);
    //上述代码会在开头插入一个100，想在第2个位置插入，只需要：
    // v.insert(v.begin()+1,100);
    // fun(v);
    //如果想向下标为1的位置插入5个100，需要：
    // v.insert(v.begin()+1,5,100);
    // fun(v);
    // vector<int> v2;
    // v2.push_back(100);
    // v2.push_back(200);
    // v2.push_back(300);
    //向vector下标为1的位置插入v2中下标为1到最后的所有元素
    // v.insert(v.begin()+1,v2.begin(),v2.end());
}

```

```

// fun(v);
//删除vector中下标为1的元素
// v.erase(v.begin() + 1);
// fun(v);
//从vector中下标为1的元素开始删除到最后
v.erase(v.begin() + 1, v.end());
fun(v);
return 0;
}

```

### 2.2.3 resize()、 swap()、 reverse()、 sort() 函数

**swap ()** 示例代码:

```

#include<bits/stdc++.h>
using namespace std;
//定义一个打印函数
void fun(vector<int> v){
    //遍历vector中的所有元素
    for(int i=0;i<v.size();i++){
        //vector中有元素时可以使用下标访问，但不可越界
        cout<<v[i]<<" ";
    }
}
int main(){
    vector<int> v;
    v.push_back(10);
    v.push_back(20);
    v.push_back(30);
    vector<int> v2;
    v2.push_back(100);
    v2.push_back(200);
    v2.push_back(300);
    cout<<"v:";
    fun(v);
    cout<<endl<<"v2:";
    fun(v2);
    swap(v,v2);
    cout<<endl<<"执行后\nv:";
    fun(v);
    cout<<endl<<"v2:";
    fun(v2);
    return 0;
}

```

### 排序、重置大小

```

#include<bits/stdc++.h>
using namespace std;
//定义一个打印函数
void fun(vector<int> v){
    //遍历vector中的所有元素
    for(int i=0;i<v.size();i++){

```

```

    //vector中有元素时可以使用下标访问，但不可越界
    cout<<v[i]<<" ";
}
}

int main(){
    int a[]={5,3,2,4,1};
    vector<int> v(a,a+5);
    sort(v.begin(),v.end());
    fun(v);
    cout<<endl;
    reverse(v.begin(),v.end());
    fun(v);
    cout<<endl;
    //重置vector的大小为20
    v.resize(20);
    fun(v);
    return 0;
}

```

## 2.2.4 二维vector

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    //外层的<>最好要有空格，否则在比较旧的编译器下无法通过
    vector< vector<int> > v(20);
    for(int i=0;i<5;i++){
        for(int j=0;j<5;j++){
            //v[i]其实是一个初始长度为0的vector
            v[i].push_back(i+j);
        }
    }

    for(int i=0;i<5;i++){
        for(int j=0;j<5;j++){
            cout<<v[i][j]<<" ";
        }
        cout<<endl;
    }
    return 0;
}

```

## 2.3 vector迭代器遍历

### 2.31.什么是迭代器

**迭代器 (iterator)** 用来指向、遍历、修改容器元素的变量，类似指针。

1. 可遍历STL容器内全部或部分元素的对象
2. 指出容器中的一个特定位置

### 2.3.2. 迭代器函数

操作	效果
*	返回当前位置上的元素值。如果该元素有成员，可以通过迭代器以 operator -> 取用
++	将迭代器前进至下一元素

前情回顾：

```
//用指针遍历数组
#include<bits/stdc++.h>
using namespace std;
int main(){
    //外层的<>最好要有空格，否则在比较旧的编译器下无法通过
    vector< vector<int> > v(20);
    char s[]{"hello,coderbee"};
    char *p;//&s $s[0]
    //p是一个指向字符数组的指针，相当于一个迭代器
    for(p=s; *p != '\0'; *p++){
        //p是指针，指向数组的不同位置， *p是元素
        cout<<*p<<" ";
    }
    return 0;
}
```

### 迭代器的定义与使用

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a[]={10,20,30,40,50};
    vector<int> v(a,a+5);
    //定义迭代器，命名为it， ::表示后面的内容属于前面，一种面向对象的写法
    vector<int>::iterator it;
    //迭代器指向vector<int>首元素
    it=v.begin();
    //修改地址上的值
    (*it)++;
    cout<<*it<<" "<<v[0]<<endl;
    //利用迭代器实现遍历
    for(it=v.begin();it<v.end();it++){
        cout<<*it<<" ";
    }
    cout<<endl;
    //利用迭代器实现反向遍历
    vector<int>::reverse_iterator rit;
    for(rit=v.rbegin();rit!=v.rend();rit++){
        cout<<*rit<<" ";
    }
    cout<<endl;

    //反着写也可以
    for(it=v.end()-1;it>=v.begin();it--){
        cout<<*it<<" ";
    }
}
```

```

        cout<<*it<<" ";
    }
    return 0;
}

```

### 2.3.3. 迭代器的分类

常用的迭代器按功能强弱氛围：**输入、输出、正向、双向、随机访问**五种，这里只讲解常见的三种。

不同容器 的迭代器，其功能的强弱也有所不同。例如，排序算法需要通过随机访问迭代器来访问容器中的元素，因此有的容器就不支持排序算法。

#### 1. 正向迭代器

1. 假设p是一个正向迭代器，则p支持以下操作：++p,p++,\*p
2. 此外，两个正向迭代器可以互相赋值，还可以用==和!=运算符进行比较

#### 2. 双向迭代器

1. 双向迭代器具备所有正向迭代器的功能
2. 双向迭代器p支持--p,p--,使得p朝着与++p相反的方向移动

#### 3. 随机访问迭代器

1. 随机访问迭代器具有双向迭代器的全部功能
2. 随机访问迭代器p还支持以下操作
  1. p+=i,使得p往后移动i个元素
  2. p-=i, 使得p向前移动i个元素
  3. p+i, 返回p后面第i个元素的迭代器
  4. p=i, 返回p后面第i个元素的叠大桥
  5. p[i], 返回p后面第i个元素的引用
  6. 两个随机访问迭代器p1、p2还可以用<,><=,>=运算符进行比较，p1<p2的含义是：p1 经过若干次（至少一次）++操作后，就会等于p2，
  7. 表达式p2-p1表示迭代器p2所指向元素和迭代器p1所指向元素序号差（p2和p1之间的元素个数减1）

### 2.3.4. 不同容器支持的迭代器

容器	迭代器类别
vector	随机
deque	随机
list	双向
set/multiset	双向
map/multimap	双向
stack	不支持迭代器

容器	迭代器类别
queue	不支持迭代器
priority_queue	不支持迭代器

## 2.4vector的应用

### 1. 【入门】数组存数

#### 题目描述

今有N个数组，初始时，N个数组均为空。共有M次操作，每次在第X个数组中加入数字Y。问最终各数组中有多少数，并将它们排序输出。

比如，输入如下数据：

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

表示有3个数组，共有5次操作，分别向第1个数组存入3，第1个数组存入2，第1个数组存入1，第2个数组存入1，第3个数组存入1。

输出如下：

```
3 1 2 3
1 1
1 1
```

第1行表示：第1个数组中有3个数，排序结果为1 2 3

第2行表示：第2个数组中有1个数，排序结果为1

第3行表示：第3个数组中有1个数，排序结果为1

#### 输入

第一行两个整数N、M ( $N \leq 100000, M \leq 300000$ )。

接下来M行，每行两个整数X、Y，含义见试题描述。 $(1 \leq X \leq N, Y \leq 10^9)$

#### 输出

共N行，第i行第一个数SUM，表示第i个数组数的个数，接下来SUM个数，为排序之后的数组。

#### 样例输入

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

#### 样例输出

```
3 1 2 3
1 1
1 1
```

思路：

## 2. 【入门】排序

### 题目描述

给定N个数组，要求先对这N个数组分别进行排序，然后再根据N的数组的字典序对这N个数组进行排序。输出排序的结果。

### 输入

第一行一个整数N，表示数组数。

接下来N ( $N \leq 1000$ ) 行，每一行先包含一个整数C ( $C \leq 1000$ )，表示数组的大小，接下来C个整数，表示数组中的一个元素。

### 输出

共N行，每行表示一个数组。

### 样例输入

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

### 样例输出

```
1
1 2
1 2 3
3
```

### 思路：

## 3. 【基础】约瑟夫问题

### 题目描述

约瑟夫问题来源于公元1世纪的犹太历史学家Josephus。问题描述，有n个人（分别以编号1, 2, 3...n表示）围成一个圆圈，从编号为1的人开始进行1 ~ m正向报数，报到m的那个人出列；他的下一个人又从1开始报数，数到m的那个人又出列；如此重复下去，直到所有的人全部出列，求最后一个出列人的编号

### 输入

输入文件仅有一行包含二个用空格隔开的整数N, M ( $2 \leq N \leq 100000, M \leq 10^9$ )。

### 输出

输出文件仅有一行包含一个整数表示一个整数，表示最后一个人在队列中的编号。

### 样例输入

```
8 3
```

### 样例输出

**思路：**

## 2.5 DTL容器的通用操作

### 2.5.1 与大小相关的操作 (size operator)

1. **size()**: 返回当前容器的元素数量
2. **empty()**: 判断容器是否为空 (true/false)
3. **max\_size()**: 返回容器能够容纳的最大元素数量

### 2.5.2 比较 (comparison)

`== , != , < , <= , > , <=`

1. 比较操作两端的容器必须属于同一类型
2. 如果两个容器内的所有元素按需相等，则两个容器相等
3. 采用字典式顺序判断某个容器是否小于另一个容器

### 2.5.3 元素操作

1. **insert(pos,e)**: 将原色e的拷贝安插在迭代器pos所指的位置
2. **erase(begin,end)**: 移出[begin,end]区间内所有的元素
3. **clear()**: 清除所有元素

### 2.5.4 赋值 (assignment) 与交换 (swap)

1. **swap()**: 用于交换元素

### 2.5.5 与迭代器 (iterator) 相关的操作 (容器需要支持迭代器)

**begin()**: 返回一个迭代器，指向第一个元素

**end()**: 返回一个迭代器，指向最后一个元素之后

**rbegin()**: 返回一个逆向迭代器，指向逆向遍历的第一个元素

**rend()**: 返回一个逆向迭代器，指向逆向遍历的最后一个元素之后

## 2.6 课后作业 (vector)

### 1. 【入门】数的操作

#### 题目描述

给定一个N个数的数组，M次操作，每次操作为下列操作之一。求最后的数组。

操作1：在第X个数之后插入一个数Y。

操作2：删除第X个数。

#### 输入

第一行两个整数N, M (N, M≤100000)，N表示数组中一开始有N个数，M表示M次操作。

第二行N个整数，表示原来的数组。

接下来M行，每行第一个数OPT，表示操作类型。

对于操作1，接下来两个数X, Y，表示在第X个数之后插入一个数Y，保证 $0 \leq X \leq$ 当前数的个数，若 $X=0$ ，表示在数组开头插入。

对于操作2，接下来一个数X，表示要删除第X个数，保证 $1 \leq X \leq$ 当前数的个数

#### 输出

输出若干个数，表示最后的数组。

#### 样例输入

```
5 3
1 2 3 4 5
1 1 6
2 1
2 2
```

#### 样例输出

```
6 3 4 5
```

### 2. 【入门】数字翻转

#### 题目描述

给定一个N个数的数组，M次操作。每次操作将数组的一段翻转，求最后的数组。（提示：可以尝试reverse函数）

#### 输入

第一行两个整数N, M (N, M≤1000) 含义见试题描述。

第二行N个整数，表示原来的数组。

接下来M行，每行两个整数X, Y (1≤X≤Y≤N)，表示翻转区间[X, Y]。

#### 输出

一行N个整数，表示操作后的数组。

#### 样例输入

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

#### 样例输出

```
1 4 3 5 2
```

### 3. 【基础】论坛帖子

#### 题目描述

你是一个论坛的站长,你们论坛有约10万个帖子,每个帖子编号为1-100000。每个帖子里面又有若干个回复。现在告诉你每个帖子下面的回复人的ID(ID的范围为1-100000)现在你要写一个程序,支持插入操作,即ADD x y 代表编号为x的帖子有一个ID为y的人回复。支持查询操作, 即QUERY x y 代表查询编号为x的帖子第y个回复的人的ID。

#### 输入

第1行一个整数N, 代表有N次询问 (1<=N<=100000)

第2行到第N+1行代表N次询问的内容, 每行为以下2种格式之一:

ADD x y 新增加了一个回复, 代表编号为x的帖子有一个ID为y的人回复

QUERY x y 代表查询编号为x的帖子第y个回复的人的ID

保证 $1 \leq x \leq 100000$ ,  $1 \leq y \leq 100000$

### 输出

对于每个QUERY的查询,每次输出占一行,代表编号为x的帖子第y个回复的人的ID,如果编号为x的帖子总的回复数小于y,则输出-1。

### 样例输入

```
8
ADD 10 10086
ADD 10 10010
QUERY 10 1
QUERY 88888 1
ADD 88888 10010
ADD 88888 12580
QUERY 88888 2
QUERY 88888 3
```

### 样例输出

```
10086
-1
12580
-1
```

## 4. 【基础】链表操作

### 题目描述

给定一个N个数的数组, M次操作, 每次操作为下列操作之一。求最后的数组。

操作1: 在第X个数之后插入一个数Y。

操作2: 删除第X个数。

操作3: 对区间[X, Y]进行排序。

操作4: 对区间[X, Y]进行翻转。

操作5: 删除区间[X, Y]中值为Z的数。

### 输入

第一行两个整数N, M (N, M≤100000) 含义见试题描述。

第二行N个整数, 表示原来的数组。

接下来M行, 每行第一个数OPT, 表示操作类型。

对于操作1, 接下来两个数X, Y, 含义见题面描述, 保证 $0 \leq X \leq$ 当前数的个数, 若 $X=0$ , 表示在数组开头插入。

对于操作2, 接下来一个数X, 含义见题面描述, 保证 $1 \leq X \leq$ 当前数的个数。

对于操作3, 接下来两个数X, Y, 含义见题面描述, 保证 $1 \leq X \leq Y \leq$ 当前数的个数, 保证操作3不超过10个。

对于操作4, 接下来两个数X, Y, 含义见题面描述, 保证 $1 \leq X \leq Y \leq$ 当前数的个数, 保证操作4不超过10个。

对于操作5, 接下来三个数X, Y, Z, 含义见题面描述, 保证 $1 \leq X \leq Y \leq$ 当前数的个数, 保证操作5不超过10个。

## 输出

输出若干个数，表示最后的数组。

## 样例输入

```
5 5
1 4 3 2 5
3 2 4
4 4 5
5 2 3 2
5 2 3 1
1 0 9
```

## 样例输出

```
9 1 3 5 4
```

### 5. 【基础】谁的孙子最多

#### 题目描述

给定一棵树，其中1号结点是根结点，问哪一个结点的孙子结点最多，有多少个。（孙子结点，就是儿子结点的儿子结点。）

#### 输入

第一行一个整数N ( $N \leq 10000$ )，表示树结点的个数。

此后N行，第i行包含一个整数Ci，表示i号结点儿子结点的个数，随后共Ci个整数，分别表示一个i号结点的儿子结点（结点编号在1~N之间）。

#### 输出

一行两个整数，表示孙子结点最多的结点，以及其孙子结点的个数，如果有多个，输出编号最小的（本题测试数据确保有解）。

## 样例输入

```
5
2 2 3
1 4
0
1 5
0
```

## 样例输出

```
1 1
```

### 6. 【基础】谁的孙子最多II

#### 题目描述

给定一棵树，其中1号结点是根结点，问哪一个结点的孙子结点最多，有多少个。（孙子结点，就是儿子结点的儿子结点。）

#### 输入

第一行一个整数N ( $N \leq 10000$ )，表示树结点的个数。此后N-1行，第i行包含一个整数 $F_i$ ，表示 $i+1$ 号结点的父亲。

### 输出

一行两个整数，表示孙子结点最多的结点，以及其孙子结点的个数，如果有多个，输出编号最小的。

### 样例输入

```
5
1
1
2
4
```

### 样例输出

```
1 1
```

## 7. 【基础】牛的视野

### 题目描述

一群高度不完全相同的牛从左到右站成一排，每头牛只能看见它右边的比它矮的牛的发型，若遇到一头高度大于或等于它的牛，则无法继续看到这头牛后面的其他牛。

给出这些牛的高度，要求每头牛可以看到的牛的数量的和。

### 输入

第一行：一个整数 $n$ ( $n \leq 10^6$ )。

第二行： $n$ 个整数，从左到右依次给出每头牛的高度,不大于 $10^6$ 。

### 输出

一行一个整数，为答案。

### 样例输入

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

### 样例输出

```
2
```

## 8. 【提高】拦截导弹方案求解

### 题目描述

某国为了防御敌国的导弹袭击，发展出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：

虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。

某天，雷达捕捉到敌国的导弹来袭。由于该系统还在试用阶段，所以只有一套系统，因此有可能不能拦截所有的导弹。

输入 $n$ 个导弹依次飞来的高度（雷达给出的高度数据是不大于 $30000$ 的正整数），计算如果要拦截所有导弹最少要配备多少套这种导弹拦截系统。

比如：有8颗导弹，飞来的高度分别为

389 207 175 300 299 170 158 165

那么需要2个系统来拦截，他们能够拦截的导弹最优解分别是：

系统1：拦截 389 207 175 170 158

系统2：拦截 300 299 165

### 输入

两行，第一行表示飞来导弹的数量n (n<=1000)

第二行表示n颗依次飞来的导弹高度 (导弹高度互不相等)

### 输出

第一行输出：要拦截所有导弹最小配备的系统数k

接下来k行分别输出每套系统拦截哪些高度的导弹

### 样例输入

```
8
389 207 175 300 299 170 158 165
```

### 样例输出

```
2
1:389 207 175 170 158
2:300 299 165
```

## 3. 双向队列 (deque)

### 3.1 什么是deque?

#### 3.1.1 deque的基本概念

deque也是顺序容器的一种，也是一个可变长度数组。要使用deque，需要包含头文件 `deque`. (也有“双端队列”的叫法)

所有适用于vector的操作都适用于deque，deque在头尾增删元素性能较好

#### 3.1.2 deque的特点

1. deque和vector有很多类似的地方。在deque中，随机存取任何元素都能在常数时间内完成（但是慢于vector）
2. 它相比于vector的优点是：vector在头部删除或添加元素的速度很慢，在尾部添加元素的性能较好，而deque在头尾增删元素都具备较好的性能（大多数情况下都能在常数时间内完成）

#### 补充知识：

1. vector本质是数组，在第十三章数组中，我们已经讨论过插入数组的过程，所以数据量越大，则效率越低
2. 所谓常数时间，是一种表达时间复杂度的概念，可以简单理解为耗时并不会因为数据多少而成指数增加

### 3.1.3 deque有两种vector没有的成员函数

1. void push\_front(const T & val); // 将val插入容器头部
2. void pop\_front(); // 删除容器头部元素

### 3.1.4 deque使用注意

1. 支持随机存取
2. 支持在头部和尾部存储数据
3. 不支持capacity和reserve操作

补充：

capacity是自动扩容，如vector中，如果容量不够，并不是一个一个添加，而是直接翻倍。比如默认为1，存第二个数据时翻倍到2，存第三个数据时并不是容量到三，而是直接变4，第五个时容量变为8，可通过vector<int> v 存入数据来验证v.capacity()的数值以证明。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    vector<int> v;
    for(int i=0;i<100;i++){
        v.push_back(i+1);
        cout<<v[i]<<" "<<v.size()<<" "<<v.capacity()<<endl;
    }
    return 0;
}
```

## 3.2 deque的应用

### 3.2.1 入门案例

#### 在头尾添加、删除函数

1. 头部添加元素：void push\_front(const T & val)
2. 头部删除元素：void pop\_front()

```
#include<bits/stdc++.h>
using namespace std;
void fun(deque<int> d){
    for(int i=0;i<d.size();i++){
        cout<<d[i]<<endl;
    }
    cout<<endl;
}
int main(){
    int a[] = {10,20,30};
    deque<int> d(a,a+3);
    fun(d);
    //向头部增加元素
    d.push_front(8);
    fun(d);
}
```

```
//删除头部元素  
d.pop_front();  
fun(d);  
//使用迭代器迭代  
cout<<"迭代器效果: "  
deque<int> :: iterator it;  
for(it=d.begin();it<d.end();it++){  
    cout<<*it<<" "  
}  
return 0;  
}
```

### 3.2.2 课后作业

#### 1. 【入门】双向队列

##### 题目描述

想想双向链表.....双向队列的定义差点儿相同，也就是说一个队列的队尾同一时候也是队首。两头都能够做出队，入队的操作。

如今给你一系列的操作。请输出最后队列的状态；

命令格式：

LIN X 表示一个整数，命令代表左边进队操作；

RIN X 表示右边进队操作；

ROUT 表示右边出队操作；

LOUT 表示从左边出队操作。

##### 输入

第一行包括一个整数M(M<=10000)，表示有M个操作；

下面M行每行包括一条命令；

命令可能不合法，对于不合法的命令，请在输出中处理；

##### 输出

输出的第一行包括队列进行了M次操作后的状态。从左往右输出，每两个之间用空格隔开。

下面若干行处理不合法的命令（假设存在）；

对于不合法的命令。请输出一行X ERROR

当中X表示是第几条命令；

##### 样例输入

```
8  
LIN 5  
RIN 6  
LIN 3  
LOUT  
ROUT  
ROUT  
ROUT  
LIN 3
```

##### 样例输出

## 4.链表 (list)

### 4.1 什么是list

#### 4.1.1 顺序存储结构的优缺点

**优点:**

1. 无需为表示结点间的逻辑关系而增加额外的存储空间
2. 可方便地随机存取表中的任一元素

**缺点:**

1. 插入或删除平均需要移动一半的结点
2. 顺序表要求占用连续的存储空间

如数组，定义时需要确定数组的大小，常见的情况就是空间浪费，或者空间不足。

#### 4.1.2 入门案例

**例子：**存储一个班同学的成绩（人数不确定），如何存储？

**解决方法一：**定义尽可能长的数组：



**解决方法二：**利用链表来存储



！！！注意！！！由于链表的特殊构造，在内存中是可以不连续的，由此也导致链表是没有下标的！这里也对应了连续存储结构优点第一条。

**补充：**

1. 链表插入数据时的过程是先让被插入的元素的下个地址指向插入位置后一个元素，然后再让插入位置前一个元素地址指向自己，最后“剪断”原位置的链接，如果先让前一个地址指向自己，相当于把原来的链表后面“剪掉”了。
2. 链表不可“回头”，因为每一个单元存储的都是下一位元素的地址，而不能找回前一个元素。由此引入“双向链表”。



#### 4.1.2 什么是list

list 是一个线性双向链表结构。它的数据由若干个结点构成，每个结点都包括一个信息块（即实际存储的数据）、一个前驱指针和一个后驱指针。它无需分配指定的内存大小且可以任意伸缩，这是因为它存储在非连续（也可以是连续）的内存空间中，并且由指针将有序的元素连接起来。

**特点：**

1. list随机检索的性能非常不好，因为它不像vector那样直接找到元素的地址，而是要从头一个个顺序查找；

2. 可以迅速在任何结点进行插入和删除操作，因为list的每个结点保存着它在链表中的位置，插入或者删除一个元素对最多三个元素有影响。
3. list支持双向迭代器，没有下标，必须使用迭代器遍历list。（由于不支持随机迭代器，所以不能写迭代器+x，迭代器-x，不可用sort()函数，但list拥有sort成员函数）

**注意：**头文件#include<list>

## 4.2 list相关函数

函数名	函数说明
push_back(元素)	往链表尾部添加元素
push_front(元素)	往链表头部添加元素
pop_back()	链表尾部移除元素
pop_front()	链表头部移除元素
insert()	在指定位置插入一个或多个元素
begin()	获取链表的起始地址
end()	获取链表的结束地址
size()	获取链表元素个数
max_size()	获取链表支持的最大容量
erase(开始地址, 结束地址)	删除链表中指定范围之间的元素
<b>remove(值val)</b>	<b>删除和val相等的元素</b>
clear()	清空链表
empty()	判断链表是否为空
reverse(开始地址, 结束地址)	翻转链表所有值
<b>sort(链表对象)</b>	<b>链表元素排序</b>

## 4.3 list的应用

```
#include<bits/stdc++.h>
using namespace std;
void fun(list<int> l){
    list<int> :: iterator it;
    for(it=l.begin();it!=l.end();it++)//不可写it<l.end()
        cout<<*it<<" ";
}
cout<<endl;
}
int main(){
    //定义list与定于vector基本一样
    // list<int> l;
    // list<int> l(5);
    int a[]={40,20,10,30,50,60};
```

```

list<int> l(a,a+6);
int x,y,i;
fun(l);

//向下标为0的位置插入15
// l.insert(l.begin(),15);
// fun(l);

//注意：双向迭代器不支持+x或者-x的写法
//插入或删除，就要从头开始移动迭代到目标位置
//获取首地址
// list<int>::iterator it=l.begin();
// cout<<"请输入要插入的位置x及要插入的值y:";
// cin>>x>>y;
// //通过迭代器获取位置
// for(i=1;i<x;i++){
//     it++;
// }
// //在it的位置插入2个y
// l.insert(it,2,y);
// fun(l);

//remove值为某数的元素
// l.remove(20);
// fun(l);

//利用成员函数sort排序
l.sort();
fun(l);

return 0;
}

```

**特别注意！** 排序时如果使用sort(l.begin(),l.end())则必然报错！因为list不支持随机迭代器，但是可以用成员函数sort()实现排序，所谓成员函数，可以简单理解为是属于list自己的函数，需要用“.”去调用，如 l.sort();

## 5.集合 (set)

### 5.1 什么是set？

set是关联容器的一种，是排序好的集合（元素已经进行了排序），**set中不能有重复元素。**

**补充：**因为不能有重复元素，所以在定义时不可以像其他容器一样定义时声明长度。

**注意：**

1. **不能直接修改set容器中元素的值。**因为元素的值被修改后，容器并不会自动重新调整顺序，于是容器的有序性就会被破坏，再在骑上进行查找等操作就会得到错误的结果。因此，如果要修改set容器中某个元素的值，正确的做法是**先删除该元素，再插入新元素。**
2. multiset容器就想set容器，但它可以保存重复的元素。
3. **set支持双向迭代器（不支持随机迭代器），在插入和删除时，要特别注意。**

4. 在STL中使用结构体，**需要对特定要求的运算符进行重载**；STL默认使用小于号来排序，因此默认重载小于号；（如果使用greater<T>比较器，就需要重载大于号），并且要注意让比较函数对相同的元素返回false。

## 5.2 set相关函数

### 5.2.1 函数介绍

函数名	函数说明
begin()	获取set容器的起始地址
end()	获取set容器的结束地址
insert()	<b>插入set容器</b>
erase(开始地址, 结束地址)	<b>删除set容器中指定范围之间的元素, set也支持直接删除值</b>
find()	<b>查找匹配的元素迭代器, 弱不存在, 返回set.end()</b>
clear()	清空set容器
size()	获取set容器元素个数
empty()	判断set容器是否为空

**注意：**如果使用set，需要引入<set>头文件，如果引入greater<T>和less<T>比较器，需要引入<functional>头文件

### 5.2.3 set的基本应用

```
#include<bits/stdc++.h>
using namespace std;
void fun(set<int> s){
    set<int>::iterator it;
    for(it=s.begin();it!=s.end();it++){
        cout<<*it<<" ";
    }
    cout<<endl;
}
int main(){
    //设定长度为0的set
//    set<int> s;
    int a[]={50,30,10,20,40};
    //利用数组初始化set
    //如果set<int,less<int>>不做指定，则默认比较器是less<T>：由小到大
//    set<int,less<int>> s(a,a+5);
//    fun(s);
    //也可以指定greater<T> 比较器，从大到小排列
    set<int,greater<int>> s(a,a+5);
    //fun(s);此时直接调用默认比较器会报错

    //迭代查看数据
//    set<int>::iterator it;
//    for(it=s.begin();it!=s.end();it++){
//        cout<<*it<<" ";
//    }
}
```

```

// }

// cout<<endl;

    //插入到特定的位置，但是插入特定位置没有意义，因为插入结束后会重新排序
// s.insert(s.begin(),60);
// s.insert(60); //插入元素60

// s.erase(s.begin()); //删除set中第一个元素

// s.erase(40); //删除值为40的元素

/*
如果使用find()查找元素的位置，返回迭代器
如果没有这个元素，则返回set.end();
*/
set<int>::iterator it;
it = s.find(500);
// *it = 300; //这里会报错，因为不可以直接修改set中的元素

if(it!=s.end()){
    cout<<*it<<"存在"<<endl;
}else{
    cout<<"元素不存在"<<endl;
}
for(it=s.begin();it!=s.end();it++){
    cout<<*it<<" ";
}
return 0;
}

```

#### 5.2.4 less<T>和greater<T>比较器及常量

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int a[]={5,2,3,4,1};
    //默认排序
// sort(a,a+5);
// sort(a,a+5,less<int>());

    //使用greater<T>比较器降序排序
    sort(a,a+5,greater<int>());
    for(int i=0;i<5;i++){
        cout<<a[i]<<" ";
    }
    cout<<endl;
    //const修饰的变量表示常量，值无法修改
    //常量一般来说要全部大写，如INT_MAX，就是系统定义的常量
    //例子
    const int X=20;
    X=30; //这里就会报错
    cout<<X;
    return 0;
}

```

### 5.2.5 set存放数据结构

```
#include<bits/stdc++.h>
using namespace std;
struct Student{
    int num;
    string name;
    int score;
    //重载operator运算符<,如果不进行处理则会报错
    bool operator>(const Student &s) const{
        //规则: 按照分数降序, 分数相同按照学号降序
        if(score>s.score||score==s.score&&num>s.num){
            return true;
        } else{
            return false;
        }
    }
};

int main(){
    set<Student,greater<Student>> s;
    Student s1 = {2,"zhao",100};
    Student s2 = {4,"qian",95};
    Student s3 = {1,"sun",95};
    Student s4 = {3,"li",95};
    s.insert(s1);
    s.insert(s2);
    s.insert(s3);
    s.insert(s4);
    set<Student>::iterator it;
    for(it=s.begin();it!=s.end();it++){
        cout<<it->num<<" "<<it->name<<" "<<it->score<<endl;
    }
    return 0;
}
```

## 5.3 课后作业 (set)

### 1. 【入门】明明的随机数

#### 题目描述

明明想在学校中请一些同学一起做一项问卷调查，为了实验的客观性，他先用计算机生成了N个1到1000之间的随机整数 ( $N \leq 100$ )，对于其中重复的数字，只保留一个，把其余相同的数去掉，不同的数对应着不同的学生的学号。然后再把这些数从小到大排序，按照排好的顺序去找同学做调查。请你协助明明完成“去重”与“排序”的工作。

#### 输入

第1行为1个正整数，表示所生成的随机数的个数：N

第2行有N个用空格隔开的正整数，为所产生的随机数。

#### 输出

第1行为1个正整数M，表示不相同的随机数的个数。第2行为M-1个用空格隔开的正整数(行尾没有多余的空格)，为从小到大排好序的不相同的随机数。

### 样例输入

```
10
20 40 32 67 40 20 89 300 400 15
```

### 样例输出

```
8
15 20 32 40 67 89 300 400
```

## 2. 【基础】统计数对个数

### 题目描述

考虑一组n个不同的正整数 $a_1, a_2, \dots, a_m$ ，它们的值在1到1000000之间。给定一个整数x。写一个程序sumx计算这样的数对个数( $a_i, a_j$ )， $1 \leq i < j \leq n$ 并且 $a_i + a_j = x$ 。

### 输入

标准输入的第一行是一个整数n( $1 \leq n \leq 1000000$ )。第二行有n个整数表示元素。第三行是一个整数x( $1 \leq x \leq 2000000$ )。

### 输出

输出一行包含一个整数表示这样的数对个数。

### 样例输入

```
9
5 12 7 10 9 1 2 3 11
13
```

### 样例输出

```
3
```

### 说明

<set>不同的和为13的数对是(12, 1), (10, 3)和(2, 11)

## 3. 【入门】等差对

### 题目描述

给定N个数 $A_i$ ，以及一个正整数C，问有多少对*i, j*，满足 $A_i - A_j = C$ 。

### 输入

第1行输入两个空格隔开的整数N和C

第2至N+1行每行包含一个整数  $A_i$

### 输出

输出一个数表示答案。

### 样例输入

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

### 样例输出

```
3
```

### 说明

$N \leq 200000$ , 所有数字保证在32位有符号整型内。

## 4. 【基础】跳房子

### 题目描述

奶牛们按不太传统的方式玩起了小孩子们玩的"跳房子"游戏。奶牛们创造了一个 $5 \times 5$ 的、由与x,y轴平行的数字组成的直线型网格，而不是用在来在里面的跳的、线性排列的、带数字的方格。然后他们熟练地在网格中的数字中跳：向前跳、向后跳、向左跳、向右跳(从不斜过来跳)，跳到网格中的另一个数字上。他们再这样跳啊跳(按相同规则)，跳到另外一个数字上(可能是已经跳过的数字)。一共在网格内跳过五次后，他们的跳跃构建了一个六位整数(可能以0开头，例如000201)。

求出所有能被这样创造出来的不同整数的总数。

### 输入

第1到5行：题面中的网格，一行5个整数

### 输出

第1行：能构建的不同整数的总数

### 样例输入

```
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 2 1
1 1 1 1 1
```

### 样例输出

```
15
```

### 说明

#### 【样例解释】

111111, 111112, 111121, 111211, 11212, 112111, 112121, 121111, 121112, 121211, 121212, 211111, 211121, 212111和 212121 能够被构建。没有其它可能的数了。

## 5. 【基础】牛的阵容

### 题目描述

农民约翰雇一个专业摄影师给他的奶牛拍照。由于约翰的牛有很多品种，他喜欢他的照片包含每个品种至少一头牛。

约翰的牛都站在数轴的不同地方，每一头牛由一个整数位置  $X_i$  以及整数品种编号  $ID_i$  表示。

约翰想拍一张照片，这张照片由数轴上连续的奶牛组成。照片的成本为这些奶牛最大和最小  $X$  坐标的差。

请帮助约翰计算最小的照片成本。保证没有两头牛在同一位置。

### 输入

第 1 行：牛的数量  $N$ ；

第  $2..1+N$  行：每行包含 2 个以空格分隔的正整数  $X_i$  和  $ID_i$ ；意义如题目描述。

### 输出

输出共一行，包含每个不同品种 ID 的照片的最低成本。

### 样例输入

```
6
25 7
26 1
15 1
22 3
20 1
30 1
```

### 样例输出

```
4
```

### 说明

对于 100% 的数据： $1 \leq N \leq 50,000$ ;  $0 \leq X_i \leq 1,000,000,000$ ;  $1 \leq ID_i \leq 1,000,000,000$ 。

样例中可选取区间[22,26]。

## 6. 映射 (map)

### 6.1 什么是pair

pair是将2个数据组合成一组数据，当需要这样的需求时就可以用pair。pair的实现是一个结构体，主要的两个成员变量是first和second。

#### 6.1.1 初始化

1. `pair<T1,T2> p1; // 创建一个空的pair对象（使用默认构造）`
2. `pair<T1,T2> p1(v1,v2); // 创建一个pair对象，使用值v1和v2初始化`
3. `make_pair(v1,v2); // 以v1和v2的值创建一个新的pair对象`

### **6.1.2 比较**

`p1 < p2` //两个pair对象间的小于运算。其定义遵循字典次序比较first和second

`p1 == p2` //如果两个对象的first和second依次相等，则这两个对象相等

### **6.1.3 访问成员变量**

`p1.first` //返回对象p1中名为first的共有数据成员

`p1.second` //返回对象p1中名为second的共有数据成员

## **6.2 什么是map**

### **6.3 常见函数**

### **6.4 map的应用**

**6.4.1 课堂练习**

**6.4.2 课后作业**

## **7.栈 (stack)**

### **7.1 什么是栈**

**7.1.1 什么是容器适配器**

**7.1.2 什么是栈**

### **7.2 栈的常见函数**

### **7.4 栈的应用**

**7.4.1 课堂练习**

**7.4.2 课后作业**

## **8.队列 (queue)**

### **8.1 什么是队列**

**8.2 queue的常见函数**

### **8.3 队列的应用**

**8.3.1 入门案例**

**8.3.2 课堂练习**

### **8.5 优先队列 (priority\_queue)**

**8.5.1 概念**

## 8.5.2 常见应用

## 8.6 课后作业

# 9.不同容器的总结

## 9.1 容器的分类

## 9.2 容器的选择

# 第三十四章 二分查找与二分答案

## 1.二分查找的相关概念

### 1.1 二分查找

二分查找也称作“折半查找 (binary search) ”，是一种效率较高的查找方法。但是折半查找要求线性表必须采用顺序存储结构，而且表中按关键字有序排列。

### 1.2 二分查找的时间复杂度 $\log_2 n$

推导：

因为二分查找每次都要排除掉一般的不合适的值，所以对n个元素的情况：

一次二分剩下 $n/2$

两次二分剩下 $n/2/2 ==> n/4$

... ...

m次二分剩下  $n/(2^m)$

在最坏的情况下是在排除到只剩下最后一个值之后得到的结果，即

$n/(2^m)=1$

所以由上式可得： $2^m=n$ ；

进而可得时间复杂度为 $\log_2 n$

注意：

$\log_2 1000000 \approx 19.9$

$\log_2 100000000 \approx 26.6$

### 1.3 二分查找常见问题

1. 二分查找元素x在数列中是否存在（求出现位置）
2. 二分查找左边界：第一次出现的位置 ( $>=$ 该元素的第一个数的位置)
3. 二分查找右边界：最后一次出现的位置 ( $>=$ 该元素的最后一个元素的位置)

## 入门案例

### 【入门】二分查找

#### 题目描述

请在一个有序递增数组中（不存在相同元素），采用二分查找，找出值x的位置，如果x在数组中不存在，请输出-1！

## 输入

第一行，一个整数n，代表数组元素个数 ( $n \leq 10^6$ )

第二行，n个数，代表数组的n个递增元素 ( $1 \leq \text{数组元素值} \leq 10^8$ )

第三行，一个整数x，代表要查找的数 ( $0 \leq x \leq 10^8$ )

## 输出

x在数组中的位置，或者-1。

## 样例输入

```
10
1 3 5 7 9 11 13 15 17 19
3
```

## 样例输出

```
2
```

## 提示

请尝试采用递归和非递归两种方式来实现二分查找

### 解法一：闭区间[l,r]

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1000100;//数据范围
int a[N],x,n;
int main(){
// cin>>n;
    scanf("%d",&n);
    for(int i = 1 ;i <= n;i++){
        scanf("%d",&a[i]);
    }
    scanf("%d",&x);
    //二分
    //交代二分的边界
    int l = 1,r = n,mid;
    //如果区间内有值[l,r]
    while(l <= r){
        mid = (l + r) / 2;//中间元素的下标
        //分情况讨论
        if(x < a[mid]) r = mid - 1;
        else if(x > a[mid]) l = mid + 1;
        else if(x == a[mid]){
            //cout<<mid;
            printf("%d",mid);
            return 0;
        }
    }
}
```

```

    }
    //找不到
    //cout<<-1;
    printf("%d", -1);
    return 0;
}

```

### 注意：求mid的三种方法

1.  $mid = (l + r) / 2;$
2.  $mid = (l + r) >> 1;$ (括号可以不写，右移一位相当于/2)
3.  $mid = l + (r - l) / 2;$ (可以防止l+r数值过大溢出的情况)

### 解法二：左闭右开区间[l,r)

```

#include<bits/stdc++.h>
using namespace std;
const int N = 1000100;//数据范围
int a[N],x,n;
int main(){
// cin>>n;
scanf("%d",&n);
for(int i = 1 ;i <= n;i++){
    scanf("%d",&a[i]);
}
scanf("%d",&x);
//二分
//交代二分的边界
//左闭右开的写法，r不在查找范围内
int l = 1,r = n + 1,mid;
//如果区间内有值[l,r)
while(l <= r){
    mid = (l + r) / 2;//中间元素的下标
    //分情况讨论
    if(x < a[mid]) r = mid ;
    else if(x > a[mid]) l = mid + 1;
    else if(x == a[mid]){
        //cout<<mid;
        printf("%d",mid);
        return 0;
    }
}
//找不到
//cout<<-1;
printf("%d", -1);
return 0;
}

```

### 注意，左闭右开区间写法的好处：

1. 上下界之差等于元素的数量
2. 易于表示两个相邻子序列，一个子序列的上界就是另一个子序列的下界

3. 表达空集时，不会使得上界小于下界

## 1.4. 二分查找边界

### 1. 【入门】二分查找左侧边界

#### 题目描述

请在一个有序不递减的数组中（数组中有相等的值），采用二分查找，找到值x第1次出现的位置，如果不存在x请输出-1。

请注意：本题要求出q个x，每个x在数组中第一次出现的位置。

比如有6个数，分别是：1 2 2 2 3 3，那么如果要求3个数：3 2 5，在数组中第一次出现的位置，答案是：5 2 -1。

#### 输入

第一行，一个整数n，代表数组元素个数 ( $n \leq 10^5$ )

第二行，n个整数，用空格隔开，代表数组的n个元素 ( $1 \leq \text{数组元素的值} \leq 10^8$ )

第三行，一个整数q，代表有要求出q个数首次出现的位置 ( $q \leq 10^5$ )

第四行，q个整数，用空格隔开，代表要找的数 ( $1 \leq \text{要找的数} \leq 10^8$ )

#### 输出

输出1行，含q个整数，按题意输出要找的每个数在数组中首次出现的位置，如果不存在这样的数，请输出-1。

#### 样例输入

```
6
1 2 2 2 3 3
3
3 2 5
```

#### 样例输出

```
5 2 -1
```

#### 解题思路

二分查找左边界注意点：

1. 二分查找，如果 $a[mid] == x$ ,还要向左侧看，判断左侧是否还是x
2. 找左界的本质是找数组中第一个 $>= x$ 的元素的位置
3. 找到位置L后，要判断 $a[L] == x$  (注意，如果都是负数，找0，要判断L在下标范围内)

#### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
const int N = 100100;
int a[N];
int n,q;//n个数，q次查询
//求元素x在数组a中首次出现的位置
int fun(int x){
    //确定边界
```

```

int l=1,r=n,mid;
while(l<=r){
    mid=l+r>>1;
    if(x<a[mid]) r = mid-1;
    else if(x>a[mid]) l = mid+1;
    else if(x==a[mid]) r = mid - 1;
}
if(a[l]==x) return l;
else return -1;
}

int main(){
    int x,i;
    cin>>n;
    for(i=1;i<=n;i++){
        cin>>a[i];
    }
    cin>>q;
    for(i=1;i<=q;i++){
        cin>>x;
        cout<<fun(x)<<" ";
    }
    return 0;
}

```

## 2. 【入门】二分查找右侧边界

### 题目描述

请在一个有序不递减的数组中（数组中的值有相等的值），采用二分查找，找到最后1次出现值x的位置，如果不存在x请输出-1。

请注意：本题要求出q个x，每个x在数组中第一次出现的位置。

比如有6个数，分别是：1 2 2 2 3 3，那么如果要求3个数：3 2 5，在数组中最后一次出现的位置，答案是：6 4 -1。

### 输入

第一行，一个整数n，代表数组元素个数 ( $n \leq 10^5$ )

第二行，n个整数，用空格隔开，代表数组的n个元素 ( $1 \leq \text{数组元素的值} \leq 10^8$ )

第三行，一个整数q，代表有要求出q个数最后一次出现的位置 ( $q \leq 10^5$ )

第四行，q个整数，用空格隔开，代表要找的数 ( $1 \leq \text{要找的数} \leq 10^8$ )

### 输出

按题意输出位置或者-1

### 样例输入

```

6
1 2 2 2 3 3
3
3 2 5

```

### 样例输出

## 解题思路

二分查找右边界注意点：

1. 二分查找，如果  $a[mid] == x$ , 还要向右侧看，判断右侧是否还是  $x$
2. 找左界的本质是找  $L$  的值，是数组中第一个  $> x$  的元素的位置
3. 因此要判断  $L-1$  (或者  $R$ ) 的值是否和  $x$  相等 ( $a[L-1] == x$ )

## 示例代码

```
#include<bits/stdc++.h>
using namespace std;
const int N = 100100;
int a[N];
int n,q;//n个数, q次查询
//求元素x在数组a中首次出现的位置
int fun(int x){
    //确定边界
    int l=1,r=n,mid;
    while(l<=r){
        mid=l+r>>1;
        if(x<a[mid]) r = mid-1;
        else if(x>a[mid]) l = mid +1;
        else if(x==a[mid]) l = mid + 1;
    }
    if(a[l-1]==x) return l-1;
    else return -1;
}
int main(){
    int x,i;
    cin>>n;
    for(i=1;i<=n;i++){
        cin>>a[i];
    }
    cin>>q;
    for(i=1;i<=q;i++){
        cin>>x;
        cout<<fun(x)<<" ";
    }
    return 0;
}
```

## 1.5.课堂练习

### 1. 【基础】同时出现的数

#### 题目描述

Medusa同学拿到了2组数字，老师请你编程帮他找出，第2组数中的哪些数，在第1组数中出现了，从小到大输出所有满足条件的数。

比如：

第1组数有： 8 7 9 8 2 6 3

第2组数有： 9 6 8 3 3 2 10

那么应该输出： 2 3 3 6 8 9

## 输入

第一行两个整数n和m，分别代表2组数的数量

第二行n个正整数

第三行m个正整数

对于60%的数据 $1 \leq n, m \leq 1000$ , 每个数 $\leq 2 \times 10^9$

对于100%的数据 $1 \leq n, m \leq 100000$ , 每个数 $\leq 2 \times 10^9$

## 输出

按照要求输出满足条件的数，数与数之间用空格隔开

### 样例输入

```
7 7
8 7 9 8 2 6 3
9 6 8 3 3 2 10
```

### 样例输出

```
2 3 3 6 8 9
```

### 解法一：二分求解

#### 思路：

第二组有哪些数在第一组中出现了，从小到大输出所有满足条件的数

采用二分法查找第二个数组中哪些数在第一个数组中出现了

1. 对a数组进行排序，方便二分查找
2. 对b数组进行排序，方便结果从小到大输出
3. 循环b数组的每个数，在a数组中二分查找是否存在

### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
const int N = 100010;
int a[N], b[N];
int n, m;
//判断是否存在
bool fun(int x){
    int l=1, r=n, mid;
    while(l <= r){
        mid=l+r>>1;
        if(x<a[mid]) r = mid-1;
        else if(x>a[mid]) l = mid+1;
        else return true;
    }
    return false;
}
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
}
```

```

for(int i=1;i<=m;i++){
    cin>>b[i];
}
sort(a+1,a+n+1);
sort(b+1,b+m+1);
for(int i=1;i<=m;i++){
    if(fun(b[i])){
        cout<<b[i]<<" ";
    }
}
return 0;
}

```

## 解法二：map解法

```

#include<bits/stdc++.h>
using namespace std;
const int N=10010;
int a[N],b[N];
int n,m;
map<int,int> ma;
int main(){
    int i;
    cin>>n>>m;
    for(i=1;i<=n;i++){
        cin>>a[i];
        ma[a[i]]=1;//标记出现过的数
    }
    for(i=1;i<=m;i++){
        cin>>b[i];
    }
    sort(b+1,b+m+1);
    for(i=1;i<=m;i++){
        if(ma[b[i]]==1){
            cout<<b[i]<<" ";
        }
    }
}
return 0;
}

```

## 2. 【基础】最满意的方案

### 题目描述

高考结束了，同学们要开始了紧张的填写志愿的过程，大家希望找一个自己最满意的大学填报方案，请你编程帮忙实现。

现有 $m (m \leq 100000)$ 所学校，每所学校预计分数线是 $a_i (a_i \leq 10^6)$ 。有 $n (n \leq 100000)$ 位学生，估分分别为 $b_i (b_i \leq 10^6)$ 。

根据 $n$ 位学生的估分情况，分别给每位学生推荐一所学校，要求学校的预计分数线和学生的估分相差最小（可高可低，毕竟是估分嘛），这个最小值为不满意度。求所有学生不满意度和的最小值。

### 输入

第一行读入两个整数m,n。m表示学校数，n表示学生数。第二行共有m个数，表示m个学校的预计录取分数。第三行有n个数，表示n个学生的估分成绩。

### 输出

一行，为最小的不满度之和。（数据保证结果<=1010）

### 样例输入

```
4 3
513 598 567 689
500 600 550
```

### 样例输出

```
32
```

### 思考：

假设某几位同学成绩为150、320、90、500，怎样求解最满意的方案？

求左边界还是右边界？

```
#include<bits/stdc++.h>
using namespace std;
const int N=100100;
int sc[N],x;
int n,m,s=0;
int main(){
    int i;
    cin>>m>>n;
    for(i=1;i<=m;i++){
        cin>>sc[i];
    }
    sort(sc+1,sc+m+1);
    int l,r,mid;
    for(i=1;i<=n;i++){
        cin>>x;
        if(x<=sc[1]) s=s+sc[1]-x;
        else if(x>=sc[m]) s=s+x-sc[m];
        else{
            l=1;
            r=m;
            while(l<=r){
                mid = l+r>>1;
                if(x<=sc[mid]) r=mid-1;
                else l=mid+1;
            }
            s=s+min(sc[l]-x,x-sc[l-1]);
        }
    }
    cout<<s;
    return 0;
}
```

## 1.6.课后作业

### 1. 【入门】二分查找满足条件的数

#### 题目描述

请在一个有序不递减的数组中（数组中的值有相等的值），采用二分查找，找到第1个大于或等于元素x的位置，如果不存在，请输出-1。

请注意：本题要求出q个x，每个x在数组找到第1个大于或等于x的元素的位置。

比如有6个数，分别是：1 2 2 2 6 6，那么如果要求3个数：5 8 2，在数组中找到第1个大于或等于他们的位置，答案是：5 -1 2。

#### 输入

第一行，一个整数n，代表数组元素个数 ( $n \leq 10^5$ )

第二行，n个整数，用空格隔开，代表数组的n个元素 ( $1 \leq \text{数组元素的值} \leq 10^8$ )

第三行，一个整数q，代表有要查询q个数 ( $q \leq 10^5$ )

第四行，q个整数，用空格隔开，代表查询的数 ( $1 \leq \text{要找的数} \leq 10^8$ )

#### 输出

按题意输出位置或者-1。

#### 样例输入

```
6
1 2 2 2 6 6
3
5 8 2
```

#### 样例输出

```
5 -1 2
```

### 2. 【入门】起止位置

#### 题目描述

有n位同学按照年龄从小到大排好队。

王老师想要查询，年龄为x的同学，在队伍中首次出现的位置和最后一次出现的位置；如果队伍中不存在年龄为x的同学，请输出-1。

由于人数太多，一个一个数，太慢啦，请你编程求解。

请注意：本题中王老师查询年龄x出现的起止位置，并不是查询了1次，而是查询了q次。

比如：

假设有6位同学的年龄为：1 2 2 2 3 3，王老师查询了4个年龄，分别是2 1 3 8，那么：

年龄为2的同学首次和最后一次出现的位置分别是：2 4；

年龄为1的同学首次和最后一次出现的位置分别是：1 1；

年龄为3的同学首次和最后一次出现的位置分别是：5 6；

年龄为8的同学首次和最后一次出现的位置分别是：-1 -1；

#### 输入

第一行包含整数n和q，表示队伍中的总人数和询问个数。

第二行包含n个整数（均在1~10000范围内），表示队伍中每个人的年龄。

接下来q行，每行包含一个整数x，表示一次询问的值。

### 输出

共q行，每行包含两个整数，表示所求年龄在队伍中的起始位置和终止位置。

如果数组中不存在该元素，则返回"-1 -1"。

### 数据范围

$1 \leq n \leq 100000$

$1 \leq q \leq 10000$

$1 \leq x \leq 10000$

### 样例输入

```
6 3
1 2 2 2 3 3
2
1
8
```

### 样例输出

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

## 3. 【提高】小X算排名

### 题目描述

小X很关心自己在学校的表现。

班主任手上有一本“个人得分记录本”，如果一位同学表现好就会加分，表现差则会扣分。学期结束，每位同学都得知了自己的个人得分。小X想知道其他同学情况如何，但由于排名不公布，他只好一个个去问班里的其他同学。

现在，小X手上有班里共N位同学的个人得分，他想知道每位同学的排名（得分相同则排名相同，见样例），可并不知道该如何计算，希望你帮帮他。

### 输入

第一行包含一个整数N。

接下来N行，第i行包含一个整数Ai，表示第i位同学的得分。

### 输出

N行，第i行包含一个整数，表示第i位同学的排名。

### 样例输入

```
5
95
100
99
99
96
```

### 样例输出

```
5  
1  
2  
2  
4
```

## 提示

### 数据范围

对于30%的数据， $N \leq 10$ 。

对于60%的数据， $N \leq 1000$ 。

对于 100%的数据， $1 \leq N \leq 100000$ ,  $0 \leq A_i \leq 100000$

## 4. 【提高】最少的修改次数

### 题目描述

现有整数  $A_1, A_2, \dots, A_n$ , 修改最少的数字为实数（整数或者小数），使得数列严格单调递增。

### 输入

第一行，一个整数  $n$ 。 ( $n \leq 10^5$ )

第二行， $n$  个整数  $A_i$ 。 ( $A_i \leq 10^9$ )

### 输出

1个整数，表示最少修改的数字的数量。

### 样例输入

```
3  
1 3 2
```

### 样例输出

```
1
```

## 5. 【提高】最长上升子序列LIS (2) ("【提高】最少的修改次数"前置题)

### 题目描述

给定一个长度为  $N$  的数列，求数值严格单调递增的子序列的长度最长是多少。

### 输入

第一行包含整数  $N$ 。

第二行包含  $N$  个整数，表示完整序列。

$1 \leq N \leq 100000$ ,  $-10^9 \leq$  数列中的数  $\leq 10^9$

### 输出

输出一个整数，表示最大长度。

### 样例输入

```
6  
1 3 2 8 5 6
```

### 样例输出

## 2.二分函数

### 1.binary\_search () : 二分查找函数

1.1.binary\_search(a+begin,a+end,x,cmp);

**函数含义：**在a数组的下标为[begin, end) 区间内，按照cmp的排序规则，找元素x，找到返回true，找不到返回false。

**注意：**

1. 查找区间是左闭右开区间， [begin, end) 不包含结束位置
2. 排序规则cmp不是必须的，但是**查找时的排序规则，必须和排序的规则是一致的**
3. 等于的含义：a等于b等价于 a在b的前面， b在a的前面都不成立

### 2.lower\_bound()和upper\_bound()

2.1 lower\_bound(): 二分查找左边界

T\* lower\_bound() (a+begin, a+end, x, cmp) :

**函数含义：**在a数组的下标为[begin, end)区间内，按照cmp的排序规则，找元素x的左边界（第一个>=元素的x的位置），返回**位置指针**；

**注意：**

1. 基本注意点同lower\_bound () ;
2. 此处返回的**不是下标的值，而是返回的指针T\* p**
3. 如果找不到符合条件的元素位置，指向下标为end的元素位置

T\* upper\_bound() (a+begin, a+end, x, cmp)

**函数含义：**在a数组的下标为[begin, end)区间内，按照cmp的排序规则，找元素x的左边界（第一个>元素的x的位置），返回**位置指针**；

**注意：**

同lower\_bound ()

### 入门案例

```
#include<bits/stdc++.h>
using namespace std;
//按个位数升序排序
bool cmp(int a1, int a2){
    return a1%10<a2%10;
}
void print(int a[], int n){
    for(int i=0;i<n;i++){
        cout<<a[i]<<" ";
    }
    cout<<endl;
}
int main(){
    int n=7;
```

```

int a[7]={12,5,3,5,98,21,7};
sort(a,a+n); //默认升序
print(a,7);
int *p = lower_bound(a,a+n,5);
cout<<*p<<" "<<p-a<<endl;
p = upper_bound(a,a+n,5);
cout<<*p<<endl;
cout<<*upper_bound(a,a+n,13)<<endl;
sort(a,a+n,cmp);
print(a,n);
cout<<*lower_bound(a,a+n,16,cmp)<<endl;
cout<<lower_bound(a,a+n,16,cmp)-a<<endl; //下标
cout<<upper_bound(a,a+n,18,cmp)-a<<endl;
if(upper_bound(a,a+n,18,cmp)==a+n){
    cout<<"Not found"<<endl;
}
cout<<*upper_bound(a,a+n,5,cmp)<<endl;
cout<<*upper_bound(a,a+n,4,cmp)<<endl;
return 0;
}

```

## 3.二分函数练习

### 1. 二分查找

#### 题目描述

请在一个有序递增数组中（不存在相同元素），采用二分查找，找出值  $xx$  的位置，如果  $xx$  在数组中不存在，请输出 -1！

#### 输入

第一行，一个整数  $n$ ，代表数组元素个数 ( $n \leq 10^6$ )

第二行， $n$  个数，代表数组的  $n$  个递增元素 ( $1 \leq 1 \leq$  数组元素值  $\leq 10^8$ )

第三行，一个整数  $x$ ，代表要查找的数 ( $0 \leq x \leq 10^8$ )

#### 输出

$xx$  在数组中的位置，或者 -1。

#### 样例输入

```

10
1 3 5 7 9 11 13 15 17 19
3

```

#### 样例输出

```

2

```

#### 说明

请尝试采用递归和非递归两种方式来实现二分查找

### 2. 二分查找左侧边界

#### 题目描述

请在一个有序不递减的数组中（数组中有相等的值），采用二分查找，找到值  $x$  第 1 次出现的位置，如果不存在  $x$  请输出 -1。

请注意：本题要求出  $q$  个  $x$ ，每个  $x$  在数组中第一次出现的位置。

比如有 6 个数，分别是：1 2 2 2 3 3，那么如果要求 3 个数：3 2 5，在数组中第一次出现的位置，答案是：5 2 -1。

### 输入

第一行，一个整数  $n$ ，代表数组元素个数 ( $n \leq 10^5$ )

第二行， $n$  个整数，用空格隔开，代表数组的  $n$  个元素 ( $1 \leq$  数组元素的值  $\leq 10^8$ )

第三行，一个整数  $q$ ，代表有要求出  $q$  个数首次出现的位置 ( $q \leq 10^5$ )

第四行， $q$  个整数，用空格隔开，代表要找的数 ( $1 \leq$  要找的数  $\leq 10^8$ )

### 输出

输出 1 行，含  $q$  个整数，按题意输出要找的每个数在数组中首次出现的位置，如果不存在这样的数，请输出 -1。

### 样例输入

```
6
1 2 2 2 3 3
3
3 2 5
```

### 样例输出

```
5 2 -1
```

### 说明

#### 【注意】

由于本题读入、输出的数据较多，C++选手请使用 `scanf` 和 `printf` 替代 `cin` 和 `cout` 提升读写效率。

## 3. 【入门】二分查找右侧边界

### 题目描述

请在一个有序不递减的数组中（数组中的值有相等的值），采用二分查找，找到最后1次出现值 $x$ 的位置，如果不存在 $x$ 请输出-1。

请注意：本题要求出 $q$ 个 $x$ ，每个 $x$ 在数组中第一次出现的位置。

比如有6个数，分别是：1 2 2 2 3 3，那么如果要求3个数：3 2 5，在数组中最后一次出现的位置，答案是：6 4 -1。

### 输入

第一行，一个整数  $n$ ，代表数组元素个数 ( $n \leq 10^5$ )

第二行， $n$  个整数，用空格隔开，代表数组的  $n$  个元素 ( $1 \leq$  数组元素的值  $\leq 10^8$ )

第三行，一个整数  $q$ ，代表有要求出  $q$  个数最后一次出现的位置 ( $q \leq 10^5$ )

第四行， $q$  个整数，用空格隔开，代表要找的数 ( $1 \leq$  要找的数  $\leq 10^8$ )

### 输出

按题意输出位置或者-1。

### 样例输入

```
6
1 2 2 2 3 3
3
3 2 5
```

### 样例输出

```
6 4 -1
```

## 4. 【基础】同时出现的数

### 题目描述

Medusa同学拿到了2组数字，老师请你编程帮他找出，第2组数中的哪些数，在第1组数中出现了，从小到大输出所有满足条件的数。

比如：

第1组数有：8 7 9 8 2 6 3

第2组数有：9 6 8 3 3 2 10

那么应该输出：2 3 3 6 8 9

### 输入

第一行两个整数n和m，分别代表2组数的数量

第二行n个正整数

第三行m个正整数

对于60%的数据 $1 \leq n, m \leq 1000$ , 每个数 $\leq 2 \times 10^9$

对于100%的数据 $1 \leq n, m \leq 100000$ , 每个数 $\leq 2 \times 10^9$

### 输出

按照要求输出满足条件的数，数与数之间用空格隔开

### 样例输入

```
7 7
8 7 9 8 2 6 3
9 6 8 3 3 2 10
```

### 样例输出

```
2 3 3 6 8 9
```

## 5. 【提高】小X算排名

### 题目描述

小X很关心自己在学校的表现。

班主任手上有一本“个人得分记录本”，如果一位同学表现好就会加分，表现差则会扣分。学期结束，每位同学都得知了自己的个人得分。小X想知道其他同学情况如何，但由于排名不公布，他只好一个个去问班里的其他同学。

现在，小X手上有班里共N位同学的个人得分，他想知道每位同学的排名（得分相同则排名相同，见样例），可并不知道该如何计算，希望你帮帮他。

### 输入

第一行包含一个整数N。

接下来N行，第i行包含一个整数Ai，表示第i位同学的得分。

### 输出

N行，第i行包含一个整数，表示第i位同学的排名。

### 样例输入

```
5
95
100
99
99
96
```

### 样例输出

```
5
1
2
2
4
```

### 提示

#### 数据范围

对于30%的数据， $N \leq 10$ 。

对于60%的数据， $N \leq 1000$ 。

对于100%的数据， $1 \leq N \leq 100000$ ,  $0 \leq A_i \leq 100000$ 。

## 6. 【基础】最满意的方案

### 题目描述

高考结束了，同学们要开始了紧张的填写志愿的过程，大家希望找一个自己最满意的大学填报方案，请你编程帮忙实现。

现有m( $m \leq 100000$ )所学校，每所学校预计分数线是ai( $a_i \leq 106$ )。有 n( $n \leq 100000$ )位学生，估分分别为 bi( $b_i \leq 106$ )。

根据n位学生的估分情况，分别给每位学生推荐一所学校，要求学校的预计分数线和学生的估分相差最小（可高可低，毕竟是估分嘛），这个最小值为不满意度。求所有学生不满意度和的最小值。

### 输入

第一行读入两个整数m,n。m表示学校数，n表示学生数。第二行共有m个数，表示m个学校的预计录取分数。第三行有n个数，表示n个学生的估分成绩。

### 输出

一行，为最小的不满度之和。（数据保证结果 $\leq 1010$ ）

### 样例输入

```
4 3
513 598 567 689
500 600 550
```

### 样例输出

## 3.二分答案

### 3.1 二分答案程序模板

二分答案与二分查找类似，也就是对有这单调性的答案进行二分，用于求解满足某种条件下的最大（小）值

**二分答案常见模板：**

```
int l=1, r=ma;
while(l<=r){
    int mid=(l+r)>>1;
    if(check(mid)) r=mid-1;
    else l=mid+1;
}
```

**补充修正：**上述模板在mid为解的时候可能会被排除在外，做出如下修正：

**二分模板一共有两个，分别适用于不同情况。**

**算法思路：**假设目标值在闭区间[l, r]中，每次将区间长度缩小一半，当l = r时，我们就找到了目标值。

**版本一：**当我们把区间[l, r]划分成[l, mid]和[mid + 1, r]时，其更新操作是r = mid或者l = mid + 1；计算mid时不需要加1。

```
int bsearch_1(int l, int r)
{
    while (l < r)
    {
        int mid = l + r >> 1;
        if (check(mid)) r = mid;
        else l = mid + 1;
    }
    return l;
}
```

**版本二：**当我们把区间[l, r]划分成[l, mid - 1]和[mid, r]时，其更新操作是r = mid - 1或者l = mid；，此时为了防止死循环，计算mid时需要加1。

```
int bsearch_2(int l, int r)
{
    while (l < r)
    {
        int mid = l + r + 1 >> 1;
        if (check(mid)) l = mid;
        else r = mid - 1;
    }
    return l;
}
```

## 3.2 课堂练习

### 1. 伐木工

#### 题目描述

伐木工人米尔科需要砍倒M米长的木材。这是一个对米尔科来说很容易的工作，因为他有一个漂亮的新伐木机，可以像野火一样砍倒森林。不过，米尔科只被允许砍倒单行树木。

米尔科的伐木机工作过程如下：米尔科设置一个高度参数H（米），伐木机升起一个巨大的锯片到高度H，并锯掉所有的树比H高的部分（当然，树木不高于H米的部分保持不变）。米尔科就行到树木被锯下的部分。

例如，如果一行树的高度分别为20, 15, 10和17，米尔科把锯片升到15米的高度，切割后树木剩下的高度将是15, 15, 10和15，而米尔科将从第1棵树得到5米，从第4棵树得到2米，共得到7米木材。

米尔科非常关注生态保护，所以他不会砍掉过多的木材。这正是他为什么尽可能高地设定伐木机锯片的原因。帮助米尔科找到伐木机锯片的最大的整数高度H，使得他能得到木材至少为M米。换句话说，如果再升高1米，则他将得不到M米木材。

#### 输入

第1行：2个整数N和M，N表示树木的数量 ( $1 \leq N \leq 10^6$ )，M表示需要的木材总长度 ( $1 \leq M \leq 2 * 10^9$ )

第2行：N个整数表示每棵树的高度，值均不超过 $10^9$ 。所有木材长度之和大于M，因此必有解。

#### 输入

1个整数，表示砍树的最高高度。

#### 样例输入

```
5 20
4 42 40 26 46
```

#### 样例输出

```
36
```

#### 思路：

本题读入的数据在int范围，但是求和可能会超过int。因此用long long来定义

思考：这是求左边界还是右边界？

在本题中，高度为mid的情况下，如果能得到 $\geq m$ 米的木材，升高高度。因此本题是求右边界

#### 示例代码

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1000010;
long long a[N];
long long n,m,l=1,r,mid;
//根据锯片高度为x的情况下，能够得到的木材量是否 $\geq m$ 
bool check(long long x){
    long long s = 0;//能够锯到的木材量
    for(int i=1;i<=n;i++){
        //如果x<树的高度，才能锯到木材
    }
}
```

```

        if(x<a[i]) s = s+a[i]-x;
        //如果得到>=m米的木材，就可以停止
        if(s>=m) return true;
    }
    return false;
}
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        cin>>a[i];
        r=max(a[i],r); //r的值应该是：所有树的最高高度
    }
    //二分答案
    while(l<=r){
        mid = l+r>>1;
        //如果高度为mid的情况下，能够得到>=m的木材
        if(check(mid)) l=mid+1;
        else r = mid -1;
    }
    cout<<l-1; //右边界
    return 0;
}

```

## 2. 防御迷阵

### 题目描述

一队士兵来到了敌军城外，准备进攻敌城。敌人在城外布置一个防御迷阵，要进入城池首先必须通过城池外的防御迷阵。

迷阵由  $n \times m$  个相同的小房间组成，每个房间与相邻四个房间之间有门可通行。而第 1 行的  $m$  个房间有  $m$  扇向外打开的门，是迷阵的入口。除了第 1 行和第  $n$  行的房间外，每个房间都安装了激光杀伤装置，将会对进入房间的人造成一定的伤害。第  $i$  行第  $j$  列造成的伤害值为  $a[i,j]$ 。（第 1 行和第  $n$  行的  $a[i,j]$  值全部为 0）。

现在士兵打算以最小伤害代价通过迷阵，显然，他们可以选择任意多的人从任意的门进入，但必须到达第  $n$  行的房间。一个士兵受到的伤害值为他在经过的路径上所有房间的伤害值中的最大值。现在，士兵们掌握了迷阵的情况，他们需要提前知道怎么安排士兵的行进路线可以使得伤害值最小。

### 输入

第一行有两个整数  $n,m$  表示迷阵的大小。

接下来  $n$  行，每行  $m$  个数，第  $i$  行第  $j$  列的数表示  $a[i,j]$ 。

数据范围： $2 \leq n,m \leq 1000$ ， $a[i,j] \leq 1000$ 。

### 输入

输出一个数，表示最小伤害代价。

### 样例输入

```

4 2
0 0
3 5
2 4
0 0

```

### 样例输出

**思路：**



第1行是入口，第n行是出口

伤害值：经过所有点的伤害值的最大

求：最小伤害代价（求极值问题）

二分答案：

1. 答案具备单调性——可以二分
2. 思考二分的范围，即答案在什么范围内
3. 思考左边界、右边界

**示例代码：**

```
#include<bits/stdc++.h>
using namespace std;
int a[1010][1010];
int n,m;
int l=INT_MAX,r=INT_MIN,mid;
//广搜变量准备
int q[1000100][3];
int fx[5]={0,0,1,0,-1};
int fy[5]={0,1,0,-1,0};
bool f[1010][1010];//标记是否走过
//判断伤害值为mid的情况下能否通过迷阵
//检验方法：采用广搜：从1,1出发，走a[i][j]<=mid的点，看能否走到第n行
bool check(int v){
    //重设标记数组的值，因为要广搜多次
    memset(f,false,sizeof(f));
    int h=1,t=1;//指针
    //交代起点
    q[1][1]=1;
    q[1][2]=1;
    f[1][1]=true;//标记走过
    int tx,ty;//要去的点
    while(h<=t){
        //遍历四个方向
        for(int i=1;i<5;i++){
            tx=q[h][1]+fx[i];
            ty=q[h][2]+fy[i];
            //如果该点可行
            if(tx>=1&&tx<=n&&ty>=1&&ty<=m&&!f[tx][ty]&&a[tx][ty]<=v){
                t++;
                q[t][1]=tx;
                q[t][2]=ty;
                f[tx][ty]=true;
                //判断是否出迷阵
                if(tx==n) return true;
            }
        }
        h++;
    }
}
```

```

    }
    return false;
}
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>a[i][j];
            //如果不是第一行和最后一行，才是伤害值
            if(i!=1&&i!=n){
                l=min(l,a[i][j]);
                r=max(r,a[i][j]);
            }
        }
    }
    //二分
    while(l<=r){
        mid = l+r>>1;
        //如果伤害值为mid，能通过，减少伤害值，再尝试
        if(check(mid)) r = mid - 1;
        else l=mid+1;
    }
    cout<<l;//左边界
    return 0;
}

```

### 3. 跳石头

#### 题目描述

一年一度的“跳石头”比赛又要开始了！

这项比赛将在一条笔直的河道中进行，河道中分布着一些巨大岩石。组委会已经选择好了两块岩石作为比赛起点和终点。在起点和终点之间，有 N 块岩石（不含起点和终点的岩石）。在比赛过程中，选手们将从起点出发，每一步跳向相邻的岩石，直至到达终点。

为了提高比赛难度，组委会计划移走一些岩石，使得选手们在比赛过程中的最短跳跃距离尽可能长。由于预算限制，组委会至多从起点和终点之间移走 M 块岩石（不能移走起点和终点的岩石）。

#### 输入

第一行包含三个整数 L,N,M，分别表示起点到终点的距离，起点和终点之间的岩石数，以及组委会至多移走的岩石数。保证 $L \geq 1$  且  $N \geq M \geq 0$ 。

接下来 N 行，每行一个整数，第 i 行的整数  $D_i$  ( $0 < D_i < L$ )，表示第 i 块岩石与起点的距离。这些岩石按与起点距离从小到大的顺序给出，且不会有两个岩石出现在同一个位置。

#### 输入

一个整数，即最短跳跃距离的最大值。

#### 样例输入

```

25 5 2
2
11
14
17
21

```

## 样例输出

4

### 说明

输入输出样例1说明：将与起点距离为 2 和 14 的两个岩石移走后,最短的跳跃距离为 4 (从与起点距离 17 的岩石跳到距离 21 的岩石,或者从距离 21 的岩石跳到终点)。

另：对于20% 的数据,  $0 \leq M \leq N \leq 10$  。

对于 50% 的数据,  $0 \leq M \leq N \leq 100$  。

对于 100% 的数据,  $0 \leq M \leq N \leq 50,000,1 \leq L \leq 1,000,000,000$  。

【来源】NOIP2015提高组复赛day2。

**思路：**直接考虑问题比较困难。我们首先考虑另一个问题，如果给定一个距离mid，问至少要移走多少势头才能满足石头之间的最小距离不小于mid？对于这个问题，答案就很简单了。我们采取贪心的策略计算：从左岸开始，移走它小于d的所有石头，再往后跳一步，循环往复。所以我们将石头按位置排序，在模拟一遍调的过程即可得到这个问题的答案。

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
const int N = 50100;
int a[N];
int n,m,l;//n个石头，搬走m个，河道长l
//求：最短跳跃距离为mid的情况下，搬走的石块数量是否<=m
bool check(int mid){
    int c=0;//搬走的数量
    int p=0;//人的位置
    //讨论所有的石块
    for(int i=1;i<=n;i++){
        //跳跃距离比mid小，不符合要求，移走石块，人不动
        if(a[i]-p<mid) c++;
        else p=a[i];//人跳过去
    }
    //判断最后一次跳跃
    if(l-p<mid) c++;
    return c<=m;
}
int main(){
    cin>>l>>n>>m;
    //读入石头
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
    //二分
    int left = 1,right = l,mid;
    while(left <= right){
        mid = left + right >> 1;
        //如果最短跳跃距离为mid的情况下，搬走的石块数量<=m，放大距离
        if(check(mid)) left = mid+1;
        else right = mid-1;
    }
    cout<<left-1;
```

```
    return 0;  
}
```

## 3.3 课后练习

### 1. 愤怒的奶牛

#### 题目描述

Farmer John 建造了一个有  $N(2 \leq N \leq 100000)$  个隔间的牛棚，这些隔间分布在一条直线上，坐标是  $x_1, x_2, \dots, x_N (0 \leq x_i \leq 1000000000)$ 。

他的  $C(2 \leq C \leq N)$  头牛不满于隔间的位置分布，它们为牛棚里其他的牛的存在而愤怒。为了防止牛之间的互相打斗，Farmer John 想把这些牛安置在指定的隔间，所有牛中相邻两头的最近距离越大越好。那么，这个最大的最近距离是多少呢？

#### 输入

第 1 行：两个用空格隔开的数字  $N$  和  $C$ 。

第 2 ~  $N+1$  行：每行一个整数，表示每个隔间的坐标。

#### 输入

输出只有一行，即相邻两头牛最大的最近距离。

#### 样例输入

```
5 3  
1  
2  
8  
4  
9
```

#### 样例输出

```
3
```

### 2. 最小的空旷指数

#### 题目描述

A市和B市之间有一条长长的高速公路，这条公路的某些地方设有路标，但是大家都感觉路标设得太少了，相邻两个路标之间往往隔着相当长的一段距离。为了便于研究这个问题，我们把公路上相邻路标的最大距离定义为该公路的“空旷指数”。

现在政府决定在公路上增设一些路标，使得公路的“空旷指数”最小。他们请求你设计一个程序计算能达到的最小值是多少。请注意，公路的起点和终点保证已设有路标，公路的长度为整数，并且原有路标和新设路标都必须距起点整数个单位距离。

#### 输入

第1行包括三个数L、N、K，分别表示公路的长度，原有路标的数量，以及最多可增设的路标数量。

第2行包括N个整数（这N个数并未排序），分别表示原有的N个路标的位置。路标的位置用距起点的距离表示，且一定位于区间[0,L]内。

**数据范围：**

50%的数据中， $2 \leq N \leq 100$ ,  $0 \leq K \leq 100$

100%的数据中， $2 \leq N \leq 100000$ ,  $0 \leq K \leq 100000$

100%的数据中， $0 < L \leq 10000000$

### 输入

输出1行，包含一个整数，表示增设路标后能达到的最小“空旷指数”值。

### 样例输入

```
101 2 1
0 101
```

### 样例输出

```
51
```

### 说明

样例说明：

公路原来只在起点和终点处有两个路标，现在允许新增一个路标，应该把新路标设在距起点50或51个单位距离处，这样能达到最小的空旷指数51。

## 第三十五章 动态规划 (DP)

### 1.什么是动态规划 (DP)

动态规划 (Dynamic Programming) 算法是解决多阶段决策过程最优的通用方法，在这类问题中，可能会有许多可行解。每一个解都对应一个只，我们希望找到具有最优值的解。

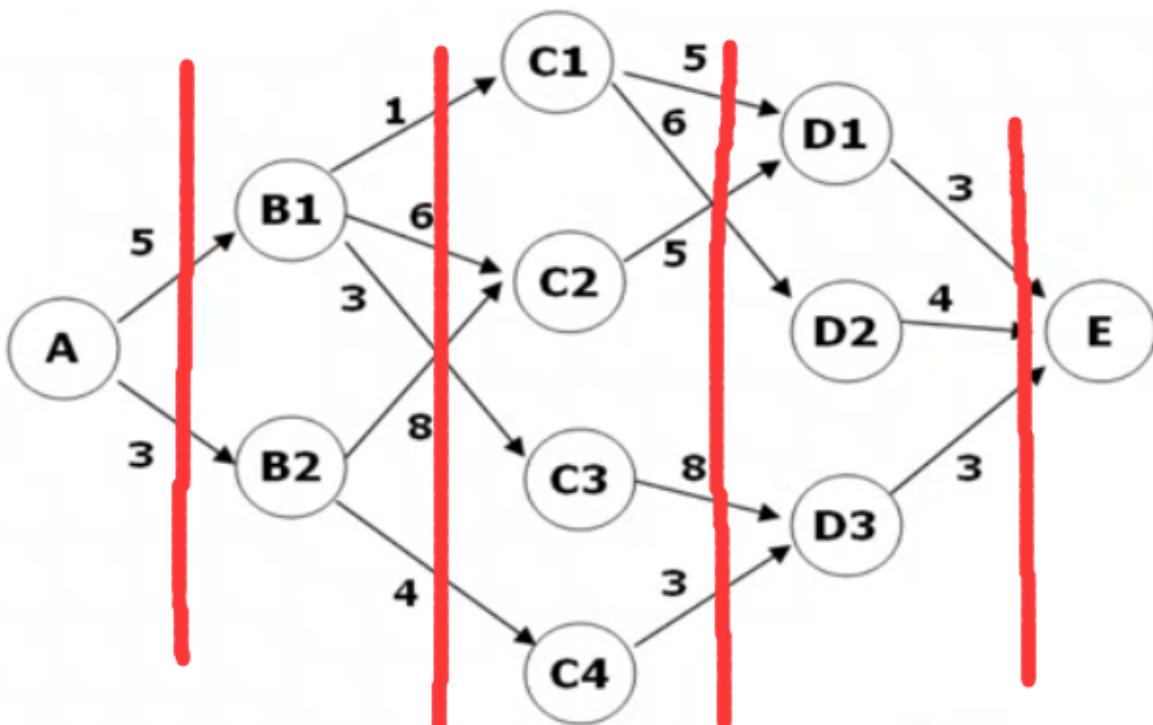
要了解动态规划的概念，首先要知道什么是多阶段决策问题

#### 多阶段决策问题

如果一类活动过程可以分为若干个相互联系的阶段，在每一个阶段都需要做出决策（采取措施），一个阶段的决策确定以后常常影响到下一个阶段的决策，从而就完全确定了一个过程的活动路线，则称它为多阶段决策问题。

各个阶段的决策构成一个决策序列，称为一个策略。每一个阶段都有若干个决策可供选择，因而就有许多策略供我们选取，对应于一个策略可以确定活动的效果，这个效果可以用数量来确定。策略不同，效果也不同，多阶段决策问题木九十要在可以选择的那些策略中间，选取一个最优策略，使得在预定的标准下达到最好的效果。

## 2.入门案例



思考：仔细观察路径的特殊性，选用搜索的方式还是贪心的方式呢？

第一阶段：A经过A-B1或者A-B2到B

第二阶段：B1有三条路径到C， B2有两条路……

解法：倒着推，设 $F(x)$ 表示x到E的最短路径长度

阶段4： $F(D_1) = 3$ ,  $F(D_2) = 4$ ,  $F(D_3) = 3$

阶段3： $F(C_1) = \min(F(D_1) + C_1 \text{ 到 } D_1 \text{ 的路径长度}, F(D_2) + C_1 \text{ 到 } D_2 \text{ 的路径长度})$

$F(x) = \min\{x \text{ 到 } y \text{ 的路径长度} + F(y)\}$  ( $y$ 是 $x$ 指向的点) —— 状态转移方程

名词解释：

1. 我们把 $F(x)$ 成为当前 $x$ 的状态
2. 每个阶段选择依赖当前的状态，有随机引起状态的转移
3. 一个决策序列 (E——D3——C4——B2——A) 就是在变化状态中产生的，故有“动态”的含义

三个基本概念

### 1. 阶段

问题的过程被分成若干个相互联系的部分，我们称为“阶段”，以便按一定的次序求解。

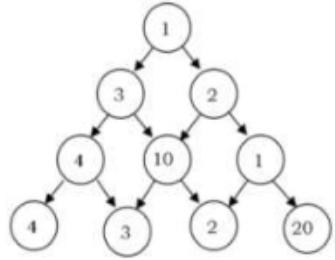
### 2. 状态

某一阶段的出发位置称为状态，通常一个阶段包含若干个状态，如第3层有 $f(C_1), f(C_2), f(C_3), f(C_4)$ 状态。

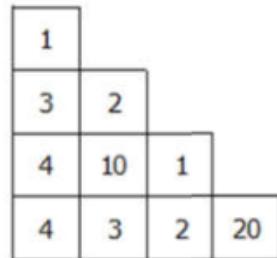
### 3. 决策

对问题的处理中做出的每种选择的行动就是决策。即从该阶段的每个状态出发，通过一次选择性的行为移至下一个阶段的相应状态

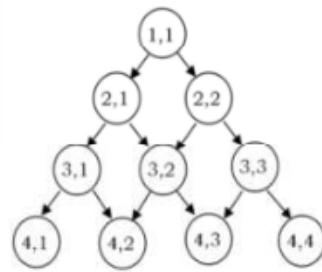
动态规划解决数塔问题的多种写法：



数字三角形



数组存储

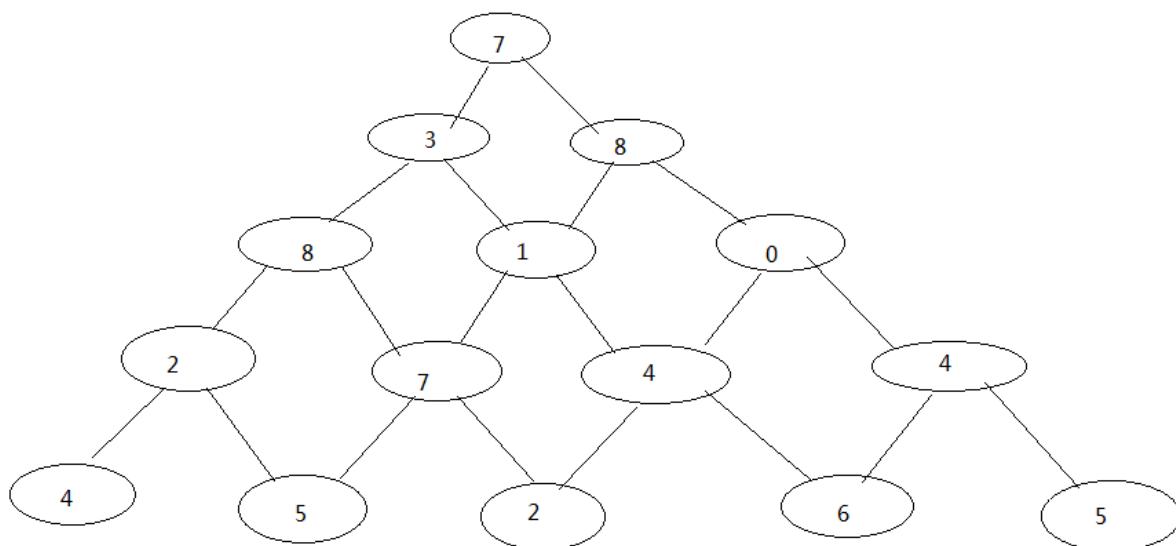


格子编号

### 例题1：数塔问题

#### 题目描述

有如下所示的数塔，要求从底层走到顶层，若每一步只能走到相邻的结点，则经过的结点的数字之和最大是多少？



#### 输入

输入数据首先包括一个整数整数N( $1 \leq N \leq 100$ )，表示数塔的高度，接下来用N行数字表示数塔，其中第*i*行有个*i*个整数，且所有的整数均在区间[0,99]内。

#### 输出

从底层走到顶层经过的数字的最大和是多少？

#### 样例输入

```
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

#### 样例输出

## 解法1：深搜示例代码

```
#include<bits/stdc++.h>
using namespace std;
int a[100][100];
int m,n;
void dfs(int x,int y,int sum){
    //程序出口
    if(x<=n){
        cout<<x<<" "<<y<<" "<<sum<<endl;
        dfs(x+1,y,sum+a[x+1][y]);
        dfs(x+1,y+1,sum+a[x+1][y+1]);
    }
}
int main(){
    int i,j;
    cin>>n;
    for(i=1;i<=n;i++){
        for(j=1;j<=i;j++){
            cin>>a[i][j];
        }
    }
    dfs(1,1,a[1][1]);
    return 0;
}
```

## 解法2：动规递推求解

### 动态规划的推导过程：

设以格子(i,j)为首的“子三角形”的最大和为d[i,j](我们将不加区别的把这个子问题 (subproblem)本身也称为d[i,j])，则问题的解是d[i,j]，我们关心的是从某处触发到地步的最大和：

1. 从(2,1)点出发的最大和记做d[2,1];
2. 从(2,2)点出发的最大和记做d[2,2];

那么从 (1,1) 点出发就有两种选择： (2,1) 或者 (2,2)；

在已知d[2,1]和d[2,2]的情况下，应该选择较大的一个。

### 考虑更一般的情况：

当前位置 (i,j) 看成一个状态，定义状态 (i,j) 的指标函数d(i,j)为从格子 (i,j)，出发时能得到的最大和（包含格子 (i,j) 本身的值），

原题的解：d(1,1)

### 思考：不同状态如何转移？

从格子(i,j)出发有两种决策：

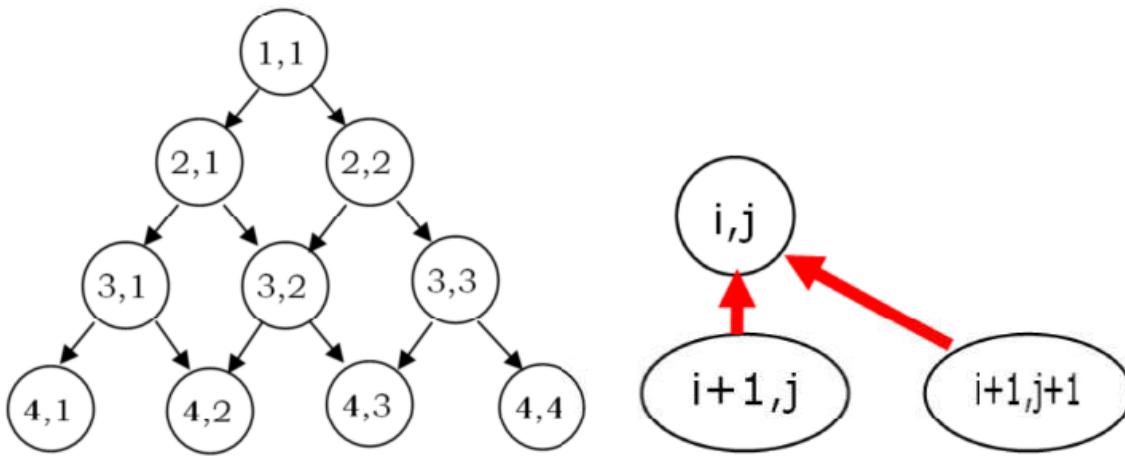
如果(i,j)格子里的值为a(i,j)

向左走需要“从(i+1,j)出发的最大和”，也就是d[i+1,j];

向右走需要“从(i+1,j+1)出发的最大和”，也就是d[i+1,j+1];

得到状态转移方程如下：

$d[i,j] = a[i,j] + \max\{d[i+1,j], d[i+1,j+1]\}$



示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int a[100][100];
int n;
int d[100][100];
void fun(){
    int i,j;
    for(j=1;j<=n;j++){
        d[n][j]=a[n][j];
    }
    for(i=n-1;i>=1;i--){
        for(j=1;j<=i;j++){
            d[i][j]=a[i][j]+max(d[i+1][j],d[i+1][j+1]);
        }
    }
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=i;j++){
            cin>>a[i][j];
        }
    }
    fun();
    cout<<d[1][1];
    return 0;
}
```

动规递归示例代码：

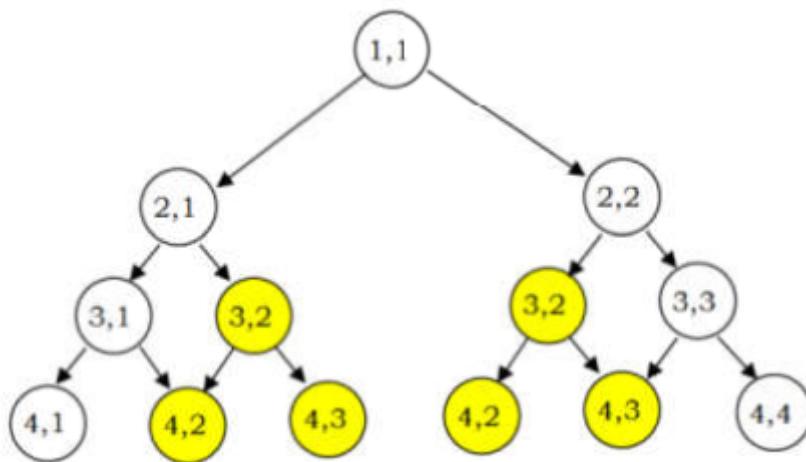
```
#include<bits/stdc++.h>
using namespace std;
int a[100][100];
```

```

int n;
//将d[i,j]=a[i,j]+max{d[i+1,j],d[i+1,j+1]} 看成一个递归
int fun(int x,int y){
    //递归的条件（出口）
    if(x<n){
        return a[x][y]+max(fun(x+1,y),fun(x+1,y+1));
    } else{
        return a[x][y];
    }
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=i;j++){
            cin>>a[i][j];
        }
    }
    cout<<fun(1,1);
    return 0;
}

```

**弊病：效率太低，因为存在大量重复计算**



**解决方案：记忆化搜索**

求解每个点的值，先判断该点的值是否已经求过，如果求解过，则直接用求过的解，否则就递归求解并保存。

记忆化搜索中用于记录值的数组要依据具体情况初始化，如本题中取值范围从0到99均可，所以可以将用于记录的d数组初始化为-1.只要不是-1就说明已经求解过了。

☆**memset(数组名, 初始化的值, 数组大小)**: 可将数组初始化，但是初始化的值只能是0或-1，其余的都不行。

**动规记忆化搜索示例代码：**

```

#include<bits/stdc++.h>
using namespace std;
int a[100][100];
int d[100][100];
int n;

```

```

/*
 *求解每个点的值，先判断该点的值是否已经求过，  

 *如果求解过，则直接用求过的解，否则就递归求解并保存  

 */
int fun(int x,int y){  

    if(x==n){  

        return a[x][y];  

    }else{  

        if(d[x][y]!=-1) {  

            return d[x][y];  

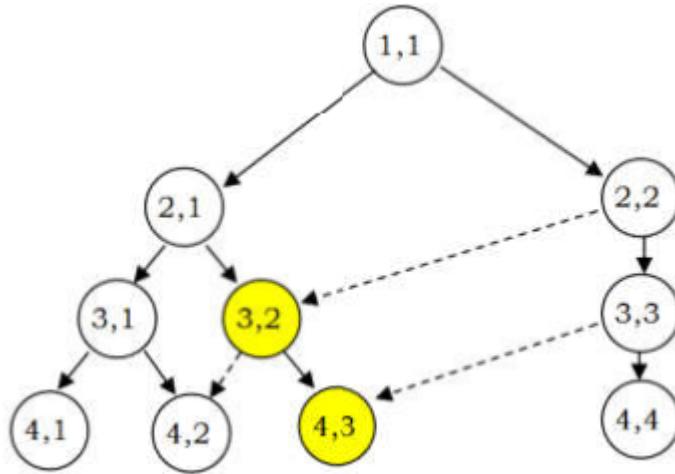
        }else{  

            d[x][y]=a[x][y]+max(fun(x+1,y),fun(x+1,y+1));  

            return d[x][y];
        }
    }
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=i;j++){
            cin>>a[i][j];
        }
    }
    memset(d,-1,sizeof(d));
    cout<<fun(1,1);
    return 0;
}

```

过程图解：

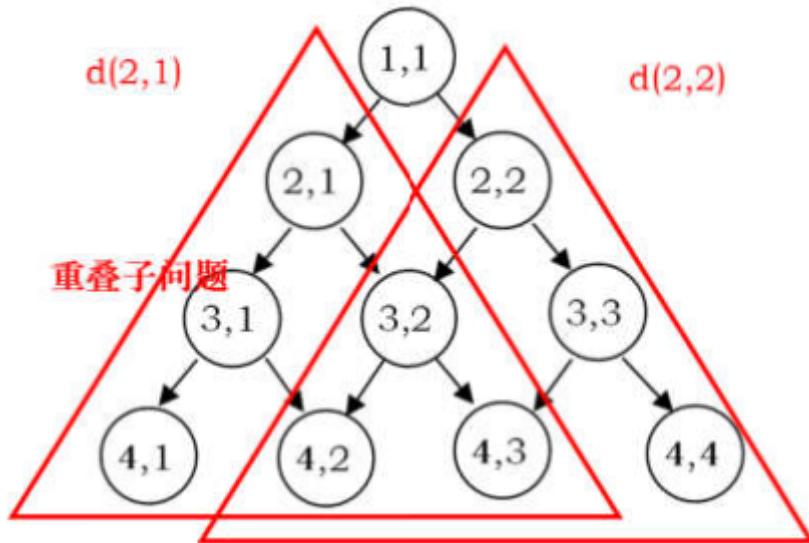


### 3. 动态规划中的基本术语

1. 动态规划的基本思想：划分阶段（子问题），确定状态及状态间的转移关系，使用递推或记忆化搜索法来实现
  1. 状态定义：用问题的某些特征参数描述一个子问题，在本题中，用 $d[i,j]$ 表示格子 $(i,j)$ 为根的子三角形的最大和。在很多时候，状态描述的细微差别将引起算法的不同。
  2. 状态转移方程：即状态值之间的递推关系。这个方程通常需要考虑两个部分，一个是递推的顺序，二是递推的边界（也可以是递推的起点）。

从直接递归和后两种方法的比较可以看出，重叠子问题（overlapping subproblems）是动态规划展示威力的关键。

**重叠子问题：** $d(1,1), d(2,1), d(2,2) \dots$ 。这些问题的共性都是从一个位置出发到底部的最大值：



2. **什么是动态规划：**动态规划并不是一种特殊的算法，而是一种针对最优化问题的一种解决途径，一种方法。

**最优化原理：**作为整个过程的最优策略具有：无论过去的状态和决策如何，对前面的决策所形成的状态而言，余下的诸决策必须构成最优策略的性质。也可以通俗的理解为子问题的局部最优解将整个问题的全局最优，而非最优解对问题的求解没有影响。

**动态规划的性质：**

1. **子问题重叠性质：**在用递归算法自顶向下对问题进行求解时，每次产生的子问题并不总是新问题，有些子问题可能被重复计算多次。动态规划算法利用此性质对每个子问题只计算一次，然后将其结果保存起来以便更高效地重（chong）用。
2. **最优化子结构性质：**若问题的最优解所包含的子问题的解也是最优的，则称该问题具有最优化子结构性质（即满足最优化原理），能用动态规划解决的求最优解问题，必须满足最优解的每个局部也都是最优的。
3. **无后效性：**即某阶段的状态一旦确定，则此后过程的演变不再受到此前各状态及决策的影响。也就是说“未来与过去无关”，当前状态是此前历史的一个完整总结。

**使用动态规划的前提：**能划分阶段，符合最优化原理，具备无后效性。

**使用动态规划求解问题的三要素：**问题的阶段，每个阶段的状态，从前一个阶段转化到后一个阶段之间的递推关系。

### 3. 动态规划的优势

1. 动态规划比穷举具有较少的计算次数

从数塔问题可以看出，层数为 $k$ 时：

1. 穷举算法求路径的条数有 $2^{k-1}$

2. 动态规划计算次数为 $\frac{k(k+1)}{2}$

2. 递归需要很大的栈空间，而动规的递推不需要栈空间

使用记忆化搜索可以有效减少计算量

### 4. 选用动态规划解决问题的基本特征

1. 动态规划一般解决最值（最大，最小，最优...）问题

2. 动态规划解决的问题是可以分阶段讨论的

3. 动态规划解决的问题必须包含最优子结构，即可以由n-1的最优解推导出n的最优解

## 5. 动态规划算法的四个步骤

1. **划分阶段**: 按照问题的时间或者空间特征，把问题分解成若干个阶段，阶段需要有序或可排序

2. **确定状态和状态变量**: 将问题发展到各个阶段时所处于的各种客观情况用不同的状态表示出来。当然，状态的选择要满足无后效性

3. **确定决策并写出状态转移方程**: 以为决策和状态转移有着天然的联系，状态转移就是根据上一阶段的状态和决策来导出本阶段的状态。所以如果确定了决策，状态转移方程也就可以写出。但事实上常常是反过来做，根据相邻两个阶段的状态之间的关系来确定决策方法和状态转移方程。

4. **构造边界条件**: 给出的状态转移方程是一个递推式，需要一个递推终止条件或者边界条件。

一般，只要解决问题的阶段、状态和状态转移决策确定了，就可以写出状态转移方程（包含边界条件）

## 6. 动态规划与分治

相同点：将问题分解成子问题，然后合并子问题的解得到原问题的解

不同的：分治的子问题**不重叠**，动态规划问题有子问题重叠。

## 7. 贪心与动态规划

相同点：要求原问题必须有最优子结构

不同点：贪心法的计算方式“自顶向下”，但并不等待子问题求解完毕后再选择使用哪一个，而是通过策略直接选择一个子问题去求解，没有被选择的子问题会被直接抛弃。这种所谓的最优选择的正确性需要用归纳法证明。而动态规划不管是采用自底向上还是自顶向下都是从边界开始向上得到目标问题的解（考虑所有子问题）

贪心：壮士断腕的决策，选了就无法后悔

动态规划：要看哪个选择笑到最后，暂时领先说明不了问题

# 4. 课堂案例

## 1. 【基础】最大部分和（连续部分和）

### 题目描述

有n个整数 ( $1 \leq n \leq 100$ )，排成一排，例如

$n=7$

-2 13 12 9 14 -10 2 (7个整数)

其最大的部分和为 48 (即  $13+12+9+14$ )

### 输入

第一行一个整数  $n$

第二行  $n$  个整数 ( $-100 \leq x_i \leq 100$ )

数之间有一个空格；其中  $x_i$  有正数

### 输出

一个整数 (即最大的连续的部分和)

### 样例输入

7

-2 13 12 9 14 -10 2

### 样例输出

### 解法1：穷举所有情况

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
/*
穷举实现，循环次数
循环次数：
n+n-1+n-2+n-3+n-4...+1
n*(n+1)/2
*/
int a[110], n, i, j;
int ma, s; //ma存放连续最大和, s存放从每个数开始连续数和
int main(){
    cin>>n;
    for(int i=1; i<=n; i++){
        cin>>a[i];
    }
    //求从每个数开始连续数的和
    for(i=1; i<=n; i++){
        s=0;
        for(j=i; j<=n; j++){
            s=s+a[j];
            ma=max(ma, s);
        }
    }
    cout<<ma;
    return 0;
}
```

### 解法2：动态规划

思路：

a 数组：存放元素

-2	13	-1	9	14	-10	2
1	2	3	4	5	6	7

dp数组：存储状态。存储以a[i]结尾的连续的最大和，最后求dp数组的最大值，就是解

-2	13	12				
1	2	3	4	5	6	7

dp[1]=a[1]=-2;

讨论以第2个数结尾连续的最大和，有2个选择：

1. 将第2个数续到前一个数的后面，形成的最大和a[2]+dp[1]=11
2. 前面的数不要了，认为连续的数只有a[2]一个，形成的最大和a[2]=13, dp[2]=13

讨论以第3个数结尾连续的最大和，也有2个选择：

1. 续:  $a[3]+dp[3]=13-1=12$

2. 不续:  $a[3]=-1$

... ...

状态转移方程总结如下:

$dp[i]=\max(dp[i-1]+a[i], a[i])$

边界:  $dp[1]=a[1]$

**示例代码:**

```
#include<bits/stdc++.h>
using namespace std;
int a[110], n, i, j, m;
int dp[110];
int main(){
    cin>>n;
    for(i=1;i<=n;i++){
        cin>>a[i];
    }
    dp[1]=a[1];
    for(i=2;i<=n;i++){
        dp[i]=max(dp[i-1]+a[i], a[i]);
        //cout<<dp[i]<<" ";
        m=max(dp[i], m);
    }
    cout<<m;
    return 0;
}
```

## 2. 【基础】最长不下降子序列 (LIS)

### 题目描述

设有由n个不相同的整数组成的数列，记为:  $a(1)、a(2)、\dots、a(n)$ 且 $a(i)<>a(j)$  ( $i<>j$ )。例如3, 18, 7, 14, 10, 12, 23, 41, 16, 24。若存在 $i_1 < i_2 < i_3 < \dots < i_e$ 且有 $a(i_1) < a(i_2) < \dots < a(i_e)$ 则称为长度为e的不下降序列。如上例中3, 18, 23, 24就是一个长度为4的不下降序列，同时也有3, 7, 10, 12, 16, 24长度为6的不下降序列。程序要求，当原数列给出之后，求出最长的不下降序列。

### 输入

第一行为n，表示n个数 ( $10 \leq n \leq 10000$ )

第二行n个整数，数值之间用一个空格分隔 ( $1 \leq a(i) \leq n$ )

### 输出

最长不下降子序列的长度

### 样例输入

```
3
1 2 3
```

### 样例输出

```
3
```

**思路：**

a 数组放元素

1	10	2	5	3	9	12	6
1	2	3	4	5	6	7	8

dp数组：存储状态，以a[i]结尾的最长不下降子序列的长度

1	2	2	3	3	4		
1	2	3	4	5	6	7	8

dp[1]=1, 以第1个数结尾的最长不下降子序列就是a[1]自己。因此有1个

dp[2]=2, 因为a[2]>a[1], 因此a[2]可以接到a[1]末尾

dp[3]=2, 因为可以续到a[1]后面, dp[3]=a[1]+1

....

也就是：当前的元素跟之前的元素挨个比较，只要能续上去就在之前的数量上加1，然后比较看跟谁续上去数字最大，谁最大就保留谁

动态转移方程为

dp[i]=1;  
dp[i]=max(dp[j]+1,dp[i]), j是从1到i直接的数, a[j]<a[i]

**示例代码：**

```
#include<bits/stdc++.h>
using namespace std;
int a[10010], dp[10010], i, j, n, mx;
int main(){
    cin>>n;
    for(i=1;i<=n;i++){
        cin>>a[i];
    }
    for(i=1;i<=n;i++){
        dp[i]=1;
        for(j=1;j<i;j++){
            if(a[i]>a[j]){
                dp[i]=max(dp[j]+1,dp[i]);
            }
        }
        mx=max(mx,dp[i]);
    }
    cout<<mx;
    return 0;
}
```

### 3. 【入门】前缀最大值

**题目描述**

求一个数列的所有前缀最大值之和。

即：给出长度为n的数列a[i],求出对于所有 $1 \leq i \leq n$ ,  $\max(a[1], a[2], \dots, a[i])$ 的和。

比如，有数列：666 304 692 188 596，前缀最大值为：666 666 692 692 692，和为3408。

对于每个位置的前缀最大值解释如下：对于第1个数666，只有一个数，一定最大；对于第2个数，求出前两个数的最大数，还是666；对于第3个数，求出前3个数的最大数是692……其余位置依次类推，最后求前缀最大值得和。

由于读入较大，数列由随机种子生成。

其中 $a[1]=x$ ,  $a[i]=(379*a[i-1]+131)\%997$ 。

### 输入

一行两个正整数n,x，分别表示数列的长度和随机种子。 $(n \leq 100000, x < 997)$

### 输出

一行一个正整数表示该数列的前缀最大值之和。

### 样例输入

```
5 666
```

### 样例输出

```
3408
```

思路：

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int a[100010], dp[100010], i, j, n, s, x;
int main(){
    cin>>n>>x;
    a[1]=x;
    for(i=2;i<=n;i++){
        a[i]=(379*a[i-1]+131)%997;
    }
    dp[1]=a[1];
    s=dp[1];
    for(i=2;i<=n;i++){
        for(j=1;j<=i;j++){
            dp[i]=max(a[i], dp[j]);
        }
        s+=dp[i];
    }
    cout<<s;
    return 0;
}
```

### 代码优化：

```
#include<bits/stdc++.h>
```

```

using namespace std;
int a[100010], dp[100010], i, j, n, s, x;
int main(){
    cin >> n >> x;
    a[1] = x;
    dp[1] = a[1];
    s = dp[1];
    for(i=2; i <= n; i++){
        a[i] = (379 * a[i-1] + 131) % 997;
        for(j=1; j <= i; j++){
            dp[i] = max(a[i], dp[j]);
        }
        s += dp[i];
    }
    cout << s;
    return 0;
}

```

#### 4. 【基础】取数

##### 题目描述

设有N个正整数 ( $1 \leq N \leq 50$ )，其中每一个均是大于等于1、小于等于300的数。

从这N个数中任取出若干个数（不能取相邻的数），要求得到一种取法，使得到的和为最大。

例如：当N=5时，有5个数分别为：13, 18, 28, 45, 21

此时，有许多种取法，如：13, 28, 21 和为62

13, 45 和为58

18, 45 和为63

.....

和为63应该是满足要求的一种取法

第一行是一个整数N

第二行有N个符合条件的整数。

一个整数，即最大和

##### 样例输出

```

5
13 18 28 45 21

```

##### 样例输出

```

63

```

##### 思路：

边界： $dp[1]=a[1]$ ,  $dp[2]=\max(a[1], a[2])$ ;

简单来说，当前的 $a[i]$ 不能加到 $dp[i-1]$ 上，因为要求不能是连续的，只能加到 $dp[i-2]$ 或者 $dp[i-3]$ 上，而此时 $dp[i-2]$ 或者 $dp[i-3]$ 已经是彼时位置最大不连续数字和了，所以可得动态转移方程为：

$dp[i]=\max(dp[i-2], dp[i-3])+a[i];$

需要注意的是，最终输出的结果应该在 $dp[n]$ 和 $dp[n-1]$ 之间选择最大值。

## 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int a[52],dp[52],i,j,n;
int main(){
    cin>>n;
    for(i=1;i<=n;i++){
        cin>>a[i];
    }
    dp[1]=a[1];
    dp[2]=a[2];
    for(i=3;i<=n;i++){
        dp[i]=max(dp[i-2],dp[i-3])+a[i];
    }
    cout<<max(dp[n],dp[n-1]);
    return 0;
}
```

## 5. 【基础】合唱队形求解

### 题目描述

N位同学站成一排，音乐老师要请其中的(N-K)位同学出列，使得剩下的K位同学不交换位置就能排成合唱队形。

合唱队形是指这样的一种队形：设K位同学从左到右依次编号为1, 2, ..., K, 他们的身高分别为T1, T2, ..., TK, 则他们的身高满足  $T_1 < T_2 < \dots < T_i, T_i > T_{i+1} > \dots > T_K (1 \leq i \leq K)$ 。

你的任务是，已知所有N位同学的身高，计算最少需要几位同学出列，可以使得剩下的同学排成合唱队形。

### 输入

输入的第一行是一个整数N ( $2 \leq N \leq 100$ )，表示同学的总数。

第一行有n个整数，用空格分隔，第i个整数  $T_i (130 \leq T_i \leq 230)$  是第i位同学的身高（厘米）。

### 输出

输出包括一行，这一行只包含一个整数，就是最少需要几位同学出列。

### 样例输入

```
8
186 186 150 200 160 130 197 220
```

### 样例输出

```
4
```

### 思路：

求最长不降序列和反过来求最长不升序列

边界：

$dp[i]=1$ ;

升的动态转移方程为：

```
if(a[j]<a[i])
dpa[i]=max(dpa[j]+1,dpa[i]);
```

降的动态转移方程为：

```
if(a[j]>a[i])
dpb[i]=max(dpb[j]+1,dpb[i]);
```

**示例代码：**

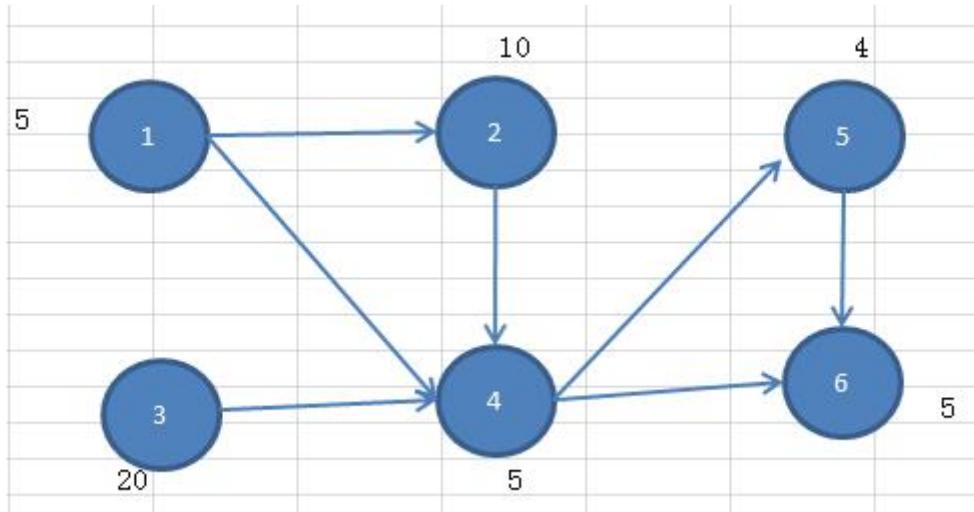
```
#include<bits/stdc++.h>
using namespace std;
int a[110],dpa[110],dpb[110],i,j,n,mx;
int main(){
    cin>>n;
    for(i=1;i<=n;i++){
        cin>>a[i];
    }
    for(i=1;i<=n;i++){
        dpa[i]=1;
        for(j=1;j<i;j++){
            if(a[i]>a[j]){
                dpa[i]=max(dpa[i],dpa[j]+1);
            }
        }
    }
    for(i=n;i>=1;i--){
        dpb[i]=1;
        for(j=n;j>i;j--){
            if(a[i]>a[j]){
                dpb[i]=max(dpb[i],dpb[j]+1);
            }
        }
    }
    //至此，如果以第i个人为中心，留下最多的人就是
    //dpa[i]+dpb[i]-1,找出最大值即可
    for(i=1;i<=n;i++){
        mx=max(mx,dpa[i]+dpb[i]-1);
    }
    cout<<n-mx;
    return 0;
}
```

## 6. 【基础】挖地雷

### 题目描述

在一个地图上有n个地窖 ( $n \leq 200$ ) ,每个地窖中埋有一定数量的地雷。同时，给出地窖之间的连接路径，并规定路径都是单向的,且保证都是小序号地窖指向大序号地窖，也不存在可以从一个地窖出发经过若干地窖后又回到原来地窖的路径。某人可以从任一处开始挖地雷，然后沿着指出的连接往下挖 (仅能选择一条路径) ，当无连接时挖地雷工作结束。设计一个挖地雷的方案，使他能挖到最多的地雷。

如下图所示：圆圈内的1 2 3 4 5 6，代表6个地窖的编号，地窖编号旁边的数字代表这个地窖地雷的数量！



### 输入

第一行：地窖的个数；

第二行为依次每个地窖地雷的个数；

下面若干行：

$x_i \ y_i$  //表示从 $x_i$ 可到 $y_i$ ,  $x_i < y_i$ 。

最后一行为"0 0"表示结束。

### 输出

第一行输出挖地雷的地窖编号的顺序: k1-k2-...-kv

第二行输出一个整数，代表最多能挖到的地雷的数量

### 样例输入

```

6
5 10 20 5 4 5
1 2
1 4
2 4
3 4
4 5
4 6
5 6
0 0

```

### 样例输出

```

3-4-5-6
34

```

### 思路：

第一个a数组存储地窖对应的地雷数

第二个dp数组存储从对应的地窖出发能挖到的最多的地雷

第三个r数组存储对应编号的地窖去了哪个编号的地窖

第四个f二维数组存储路径通断情况。

与求数字塔思路类似。动态转移方程：

$dp[i] = a[i] + \max(dp[j]) \quad j=i+1 \dots n$  ij有通路

$dp[n] = a[n]$

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int n, i, j, x, y;
int a[210], dp[210], r[210];
bool f[200][200];
int main(){
    cin >> n;
    for(i=1; i <= n; i++){
        cin >> a[i];
    }
    while(1){
        cin >> x >> y;
        if(x == 0 && y == 0) break;
        f[x][y] = true;
    }
    dp[n] = a[n];
    //index表示要去的位置
    //count表示当前最多的地雷数
    int index, count = 0;
    for(i=n-1; i >= 1; i--){
        for(j=i+1; j <= n; j++){
            if(f[i][j] && dp[j] > count){
                count = dp[j];
                index = j;
            }
        }
        dp[i] = a[i] + count;
        r[i] = index;
    }
    index = 1;
    int ans = dp[index];
    for(i=2; i <= n; i++){
        if(dp[i] > ans){
            index = i;
        }
    }
    ans = dp[index];
    while(index != 0){
        cout << index;
        index = r[index];
        if(index != 0) cout << "-";
    }
    cout << endl;
    cout << ans;
    return 0;
}
```

## 7. 【入门】简单背包问题

题目描述

有一个背包能装的重量maxw(正整数,  $0 \leq maxw \leq 20000$ )，同时有n件物品( $0 \leq n \leq 100$ )，每件物品有一个重量wi(正整数)和一个价值pi(正整数)。要求从这n件物品中任取若干件装入背包内，使背包的物品价值最大。

第1行：背包最大载重maxw，物品总数n

第2行到第n+1行：每个物品的重量和价值

一个数字即背包内物品最大价值

#### 样例输出

```
10 3  
4 5  
3 4  
6 9
```

#### 样例输出

```
14
```

思路：

示例代码：

## 3.课后作业

### 1. 【入门】跳格子

#### 题目描述

地面上有一排长度为n的格子1-n，每个格子上都有一个数xi，开始时你在位置0，每次你可以向前跳1-2格，然后取走格子上的数，直到跳到位置n+1。取走的数的和就是你的得分，现在你想知道你可能的最大得分是多少。

一行四个整数n,A,B,C( $n \leq 100000, 0 \leq A, B, C \leq 10000$ )，其中n表示格子的数量。 $x[i]$ 由如下方式生成：

```
for (int i = 1; i <= n; i++){  
    int tmp = ((long long)A * i * i + B * i + C) % 20000;  
    x[i] = tmp - 10000;  
}
```

一行一个整数ans表示可能的最大得分。

#### 样例输出

```
3 1 1 1
```

#### 样例输出

```
-9993
```

### 2. 【入门】跳格子2

### 题目描述

地面上有一排长度为n的格子1-n, 每个格子上都有一个数 $x_i$ , 开始时你在位置0, 每次你可以向前跳1-2格, 然后取走格子上的数, 直到跳到位置n+1。取走的数的和就是你的得分, 现在你想知道你可能的最小得分是多少。

一行四个整数n,A,B,C( $n \leq 100000, 0 \leq A, B, C \leq 10000$ ), 其中n表示格子的数量。 $x[i]$ 由如下方式生成:

```
for (int i = 1; i <= n; i++){
    int tmp = ((long long)A * i * i + B * i + C) % 20000;
    x[i] = tmp - 10000;
}
```

一行一个整数ans表示可能的最小得分。

### 样例输出

```
3 1 1 1
```

### 样例输出

```
-29977
```

## 3. 【入门】前缀最小值

### 题目描述

求一个数列的所有前缀最小值之和。

即: 给出长度为n的数列 $a[i]$ , 求出对于所有 $1 \leq i \leq n$ ,  $\min(a[1], a[2], \dots, a[i])$ 的和。

由于读入较大, 数列由随机种子生成。

其中 $a[1]=x$ ,  $a[i]=(379*a[i-1]+131)\%997$ 。

一行两个正整数n,x, 分别表示数列的长度和随机种子。 $(n \leq 100000, x < 997)$

一行一个正整数表示该数列的前缀最小值之和。

### 样例输出

```
5 666
```

### 样例输出

```
1650
```

## 4. 【基础】装箱问题

### 题目描述

有一个箱子容量为V (正整数,  $0 \leq V \leq 200000$ ), 同时有n个物品 ( $0 < n \leq 30$ ), 每个物品有一个体积 (正整数)。

要求n个物品中, 任取若干个装入箱内, 使箱子的剩余空间为最小。

1个整数, 表示箱子容量

1个整数, 表示有nnn个物品

接下来n行, 分别表示这n个物品的各自体积

1个整数, 表示箱子剩余空间。

### 样例输出

```
24  
6  
8  
3  
12  
7  
9  
7
```

### 样例输出

```
0
```

## 5. 【提高】吃鱼 (fish)

### 题目描述

小花爱吃鱼，这是全世界都知道的事情。它的好朋友编程兔给它准备了很多的零食，每一样都是小花喜欢的。当然了，里面最多的肯定是鱼。某一天编程兔给小花准备了两种鱼，一种鱼的重量是 1，另一种鱼的重量是 2，重量为 1 的鱼有不同的美味值，重量为 2 的鱼也有不同的美味值。现在假设小花的胃口最多能吃下不超过重量为  $v$  的鱼，小花希望吃掉的鱼的美味值总和最大。

输入数据第一行是两个正整数  $n$  和  $v$ ， $n$  表示鱼的数量， $v$  表示小花的胃口。接下来  $n$  行，每行两个正整数，第一个正整数表示鱼的重量（只有 1 和 2 两种可能），另一个正整数表示这条的美味值。

输出只有一行一个整数，表示小花能得到的最大美味值总和。

### 样例输出

```
3 2  
1 2  
2 7  
1 3
```

### 样例输出

```
7
```

## 6. 【基础】采药

### 题目描述

辰辰是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”

如果你是辰辰，你能完成这个任务吗？

第一行有 222 个整数  $T(1 \leq T \leq 1000)T(1 \leq T \leq 1000)T(1 \leq T \leq 1000)$  和  $M(1 \leq M \leq 100)M(1 \leq M \leq 100)M(1 \leq M \leq 100)$ ，用一个空格隔开， $T$  代表总共能够用来采药的时间， $M$  代表山洞里的草药的数目。

接下来的  $M$  行每行包括两个在 111 到 100100100 之间（包括 111 和 100100100）的整数，分别表示采摘某株草药的时间和这株草药的价值。

111 个整数，表示在规定的时间内可以采到的草药的最大总价值。

### 样例输出

```
70 3
71 100
69 1
1 2
```

### 样例输出

```
3
```

## 7. 【提高】机器分配

### 题目描述

总公司拥有高效设备M台，准备分给下属的N个分公司。各分公司若获得这些设备，可以为国家提供一定的盈利。问：如何分配这M台设备才能使国家得到的盈利最大？求出最大盈利值。

其中 $M \leq 15$ ,  $N \leq 10$ 。分配原则：每个公司有权获得任意数目的设备，但总台数不超过设备数M。

第一行有两个数，第一个数是分公司数N，第二个数是设备台数M。

接下来是一个 $N \times M$ 的矩阵，表明了第i个公司分配j台机器的盈利。

第一行为一个整数，代表最大盈利的值。

接下来N行，每行两个数，用空格隔开，第一个数代表了第i个公司的序号，第二个数代表该公司分得机器的数量。

### 样例输出

```
3 3
30 40 50
20 30 50
20 25 30
```

### 样例输出

```
70
1 1
2 1
3 1
```

## 第二十六章 动态规划的子序列及背包问题进阶

# 码蜂C++程序设计--数据结构与进阶算法

## 第一章 并查集

### 1.什么是并查集

并查集是一种树型的数据结构，用于处理一些不相交集合的合并及查询问题。

并查集的思想是用一个数组表示了整片森林（parent），树的根节点唯一标识了一个集合，我们只要找到某个元素的树根，就能确定它在哪个集合里。

## 第二章 图论基础

## 第三章 图论--最短路径

## 第四章 背包DP

### 1.入门问题

#### 1. 【提高】简单背包问题

##### 题目描述

有一个背包能装的重量 $\text{maxw}$ (正整数,  $0 \leq \text{maxw} \leq 20000$ ), 同时有 $n$ 件物品( $0 \leq n \leq 100$ ), 每件物品有一个重量 $w_i$ (正整数)和一个价值 $v_i$ (正整数)。要求从这 $n$ 件物品中任取若干件装入背包内, 使背包的物品价值最大。

##### 输入

第1行: 背包最大载重 $\text{maxw}$ , 物品总数 $n$

第2行到第 $n+1$ 行: 每个物品的重量和价值

##### 输出

一个数字即背包内物品最大价值

##### 样例输入

```
10 3
4 5
3 4
6 9
```

##### 样例输出

```
14
```



修改数组:



动态转移方程:  $dp[i][j] = \max(dp[i-1][j], v[i] + dp[i-1][j - w[i]])$

##### 示例代码:

```
#include<bits/stdc++.h>
using namespace std;
//状态转移方程:
//dp[i][j] = max(dp[i-1][j] , v[i] + dp[i-1][j-w[i]])
//c:背包容量
//dp[i][j]:有i件物品, 背包容量为j的情况下存储的最大价值
int c,dp[110][20100],w[110],v[110],i,j,n;
int main(){
    cin>>c>>n;
    for(i=1;i<=n;i++){
        cin>>w[i]>>v[i];
    }
    //递推求dp数组
```

```

//i表示物品数量
for(i=1;i<=n;i++){
    //在i件物品，讨论背包容量分别是1~c的情况下最大价值
    //j表示背包容量
    for(j=1;j<=c;j++){
        //如果放得下
        if(w[i]<=j){
            dp[i][j]=max(v[i]+dp[i-1][j-w[i]],dp[i-1][j]);
        }else{//如果装不下
            dp[i][j]=dp[i-1][j];
        }
    }
}
cout<<dp[n][c];
return 0;
}

```

## 2. 【提高】简单背包问题

### 题目描述

有一个背包能装的重量 $\text{maxw}$ (正整数,  $0 \leq \text{maxw} \leq 20000$ ), 同时有 $n$ 件物品( $0 \leq n \leq 100$ ), 每件物品有一个重量 $w_i$ (正整数)和一个价值 $v_i$ (正整数)。要求从这 $n$ 件物品中任取若干件装入背包内, 使背包的物品价值最大。

### 输入

第1行: 背包最大载重 $\text{maxw}$ , 物品总数 $n$

第2行到第 $n+1$ 行: 每个物品的重量和价值

### 输出

一个数字即背包内物品最大价值

### 样例输入

```

10 3
4 5
3 4
6 9

```

### 样例输出

14

**问题描述:** 有 $n$ 件物品和容量为 $m$ 的背包, 给出 $i$ 件物品的种类及价值, 求解让装入背包的物品重量不超过背包容量且价值最大。

**特点:** 最简单的背包问题, 特点是每个物品只有一件供你选择放还是不放, 所以此类问题又叫做“01”背包。

### 一维解法:

$dp[j] = \max(dp[j], dp[j-wi] + vi);$

### 示例代码:

```

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

```

```

//二维数组状态转移方程:
//dp[i][j] = max(dp[i-1][j] , v[i] + dp[i-1][j-w[i]])
//一维数组状态转移方程
//dp[j]=max(dp[j],dp[j-wi]+vi);
int i,j,wi,vi,dp[20010],n,c;
int main(){
    cin>>c>>n;
    for(i=1;i<=n;i++){
        cin>>wi>>vi;
        for(j=c;j>=wi;j--){
            dp[j]=max(dp[j],vi+dp[j-wi]);
        }
    }
    cout<<dp[c];
    return 0;
}

```

## 2.完全背包问题

N件物品，容量为W的背包，第i件物品的重量为Wi，价值为Vi，每件物品有无数个，求能装的最大价值。

### 1. 采灵芝

#### 题目描述

仙岛上种了无数的不同种类的灵芝，小芳跟着爷爷来到仙岛采摘灵芝。由于他们带的食物和饮用水有限，必须在时间 t 内完成采摘。

假设岛上有 m 种不同种类的灵芝，每种灵芝都有无限多个，已知每种灵芝采摘需要的时间，以及这种灵芝的价值；

请你编程帮助小芳计算，在有限的时间 t 内，能够采摘到的灵芝的最大价值是多少？

#### 输入

输入第一行有两个整数 T (1≤T≤100000) 和 M (1≤M≤2000)，用一个空格隔开，T代表总共能够用来采灵芝的时间，M 代表岛上灵芝的种类数。

接下来的 M 行每行包括两个在 1 到 10000 之间（包括 1 和 10000）的整数，分别表示采摘某种灵芝的时间和这种灵芝的价值。

#### 输出

输出一行，这一行只包含一个整数，表示在规定的时间内，可以采到的灵芝的最大总价值。

#### 样例输入

```

70 3
71 100
69 1
1 2

```

#### 样例输出

140

#### 思路：

01背包：

二维数组：

$dp[i][j] = \max(dp[i-1][j], dp[i-1][j-w[i]] + v[i])$

滚动优化：从背包容量c降序循环到当前物品重量wi

$dp[j] = \max(dp[j], d[j-wi] + vi)$

完全背包：每种物品有无限多个(注：后面k范围： $1 \leq k \leq w/w[i]$ ，因为k为0的状态在前面)

1.  $dp[i][j] = \max(dp[i-1][j], dp[i-1][j-k*w[i]] + k*v[i])$

上述方程加上k=0的状态（即不去第i件物品）时，可以变成下面这样：

2.  $dp[i][j] = \max(dp[i-1][j-k*w[i]] + k*v[i])$  (注：k范围： $0 \leq k \leq w/w[i]$ )

把k=0的情况单独考虑，即比较 **不放第i件物品、放第i件物品k件 ( $k \geq 1$ )两个结果中最大的那个k**这两种情况下谁的结果更大

3.  $dp[i][j] = \max(dp[i-1][j], \max(dp[i-1][j-k*w[i]] + k*v[i]))$  注：  $k \geq 1$

上面这个方程中，先把装了k件的第i件物品的价值与不装第i件物品的最大值进行了比较，但k的取值范围至少要是1，如果k为0，则max中两个方程的就成了相同的状态了。

考虑到上述方程中**放第i件物品**的情况，放的话至少要一件，如果将方程变化为可以放0件的情况，方程可以变形为：

4.  $dp[i][j] = \max(dp[i-1][j], \max(dp[i-1][(j-w[i])-k*w[i]] + k*v[i]) + v[i])$

此时，k的值可以从0开始了，即 $k \geq 0$ 。这时，我们将方程2拿出来，将其中的j换成 $j-w[i]$ ，即可得到方程5：

5.  $dp[i][j-w[i]] = \max(dp[i-1][j-w[i]-k*w[i]] + k*v[i])$

此时，可以发现，方程5等号后面的内容与方程4的max中的后半段完全一样，简单等量代换可以得到：

$dp[i][j] = \max(dp[i-1][j], dp[i][j-w[i]] + v[i]);$

**解法一：二维数组，但需要注意内存是否超限**

```
#include<bits/stdc++.h>
using namespace std;
int n,maxw;
int dp[1010][1010],i,j;
int w[10100],v[10100];
int main(){
    cin>>maxw>>n;
    for(i=1;i<=n;i++){
        cin>>w[i]>>v[i];
    }
    for(i=1;i<=n;i++){
        for(j=1;j<=maxw;j++){
            if(j>=w[i]){
                dp[i][j]=max(dp[i-1][j],dp[i][j-w[i]]+v[i]);
            }else{
                dp[i][j]=dp[i-1][j];
            }
        }
    }
    cout<<dp[n][maxw];
    return 0;
}
```

## 解法二：一维数组优化，注意此时需要顺推

注意观察完全背包的状态转移方程与01背包状态转移方程的异同：

01背包： $dp[i][j] = \max(dp[i-1][j], dp[i-1][j-w[i]] + v[i])$

完全背包： $dp[i][j] = \max(dp[i-1][j], dp[i][j-w[i]] + v[i])$ ;

经过比对我们很容易发现，在01背包问题中，我们需要得到上一行的状态，即*i-1*行，如果顺推，则会产生值被覆盖的问题，所以需要逆推；

而在完全背包问题下，我们需要知道的是同一行的状态，即*i*，所以必须从当前物品重量*wi*开始顺推循环到背包容量，易得，状态转移方程如下：

$dp[j] = \max(dp[j], dp[j-wi] + vi)$

```
#include<bits/stdc++.h>
using namespace std;
int n,maxw;
int dp[100100],i,j;
int wi,vi;
int main(){
    cin>>maxw>>n;
    for(i=1;i<=n;i++){
        cin>>wi>>vi;
        for(j=wi;j<=maxw;j++){
            if(j>=wi){
                dp[j]=max(dp[j],dp[j-wi]+vi);
            }
        }
    }
    cout<<dp[maxw];
    return 0;
}
```

## 3.多重背包问题

### 1. 多重背包（1）

#### 题目描述

有 N 种物品和一个容量是 V 的背包。

第 i 种物品最多有  $s_i$  件，每件体积是  $v_i$ ，价值是  $w_i$ 。

求解将哪些物品装入背包，可使物品体积总和不超过背包容量，且价值总和最大。

输出最大价值。

#### 输入

第一行两个整数，N, V，用空格隔开，分别表示物品种数和背包容积。

接下来有 NN 行，每行三个整数  $v_i, w_i, s_i$ ，用空格隔开，分别表示第 i 种物品的体积、价值和数量。

数据范围：

$0 < N, V \leq 100, 0 < v_i, w_i, s_i \leq 100$ 。

#### 输出

输出一个整数，代表最大价值。

## 样例输入

```
4 10
3 2 2
4 3 2
2 2 1
5 3 4
```

## 样例输出

```
8
```

### 思路：

解法一：将多重背包的 $s_i$ 个物品分别装入 $w$ 和 $v$ 数组，直接转换为01背包

01背包：每种物品有1件

完全背包：每种物品有无限件

多重背包：每种物品有 $s_i$ 件

将 $s_i$ 件物品都存起来，转化为有 $s_i$ 个物品，每个物品有1件

```
#include<bits/stdc++.h>
using namespace std;
int n,c,i,j;
int v[10010],w[10010];
int dp[110];
int vi,wi,si,k;//k代表数组下标
int main(){
    cin>>n>>c;
    for(i=1;i<=n;i++){
        cin>>vi>>wi>>si;
        //第i个物品有 $s_i$ 件，都存入数组
        for(j=1;j<=si;j++){
            k++;
            v[k]=vi;
            w[k]=wi;
        }
    }
    //01背包解法
    for(i=1;i<=k;i++){
        for(j=c;j>=v[i];j--){
            dp[j]=max(dp[j],dp[j-v[i]]+w[i]);
        }
    }
    cout<<dp[c];
    return 0;
}
```

解法二：在做01背包时，体现一下 $s_i$ 件物品这个条件

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

```

int n,c,i,j;
int v[110],w[110],s[110];
int dp[110];
int vi,wi,si,k;
int main(){
    cin>>n>>c;
    for(i=1;i<=n;i++){
        cin>>v[i]>>w[i]>>s[i];
    }
    //01背包
    for(i=1;i<=n;i++){
        for(k=1;k<=s[i];k++){
            //逆序从背包容量循环到当前物品体积
            for(j=c;j>=v[i];j--){
                dp[j]=max(dp[j],dp[j-v[i]]+w[i]);
            }
        }
    }
    cout<<dp[c];
    return 0;
}

```

## 2. 多重背包 (2)

### 题目描述

有  $N$  种物品和一个容量是  $V$  的背包。

第  $i$  种物品最多有  $s_i$  件，每件体积是  $v_i$ ，价值是  $w_i$ 。

求解将哪些物品装入背包，可使物品体积总和不超过背包容量，且价值总和最大。

输出最大价值。

### 输入

第一行两个整数， $N$ ,  $V$ ，用空格隔开，分别表示物品种数和背包容积。

接下来有  $N$  行，每行三个整数  $v_i, w_i, s_i$ ，用空格隔开，分别表示第  $i$  种物品的体积、价值和数量。

### 输出

输出一个整数，表示最大价值。

数据范围：

$0 < N \leq 10000$ ,  $0 < V \leq 2000$ ,  $0 < v_i, w_i, s_i \leq 20000$

### 样例输入

```

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

```

### 样例输出

## 4.混合背包

### 4.1 课堂案例

#### 1. 混合背包

##### 题目描述

有  $N$  种物品和一个容量是  $V$  的背包。

物品一共有三类：

1. 第一类物品只能用 1 次（01 背包）；
2. 第二类物品可以用无限次（完全背包）；
3. 第三类物品最多只能用  $s_i$  次（多重背包）；

每种体积是  $v_i$ ，价值是  $w_i$ 。

求解将哪些物品装入背包，可使物品体积总和不超过背包容量，且价值总和最大。

输出最大价值。

##### 输入

第一行两个整数， $N$ ， $V$ ，用空格隔开，分别表示物品种数和背包容积。

接下来有  $NN$  行，每行三个整数  $v_i, w_i, s_i$  用空格隔开，分别表示第  $i$  种物品的体积、价值和数量。

1.  $s_i = -1$  表示第  $i$  种物品只能用 1 次；
2.  $s_i = 0$  表示第  $i$  种物品可以用无限次；
3.  $s_i > 0$  表示第  $i$  种物品可以使用  $s_i$  次；

##### 数据范围

$0 < N, V \leq 10000$ ,  $0 < v_i, w_i \leq 10000$ ,  $-1 \leq s_i \leq 1000$ 。

##### 输出

输出一个整数，表示最大价值。

##### 样例输入

```
4 5
1 2 -1
2 4 1
3 4 0
4 5 2
```

##### 样例输出

```
8
```

### 4.2 课后练习

#### 1. 环游世界之背包问题

##### 题目描述

张老师准备环游世界，出发之前要做的最重要的事情，当然是整理自己的背包啦。张老师有一个容积为m的背包，有n个物品作为放入背包的待选物品，每样东西都有自己的价值Wi，和体积Vi，第i个物品有Ni个（Ni=0时表示有无限多个），请你编程帮助张老师计算，他的背包能够存入的最大价值是多少？

30%数据满足  $1 \leq m, n \leq 1000$

100%数据满足  $1 \leq m, n \leq 10000$

### 输入

第1行有2个整数N,M，表示物品的种类和背包的容积；

第2-N+1行，每行有3个整数Vi, Wi, Pi，分别表示每个物品的体积，价值，个数。（体积 $\leq 5000$ ，价值 $\leq 5000$ ，个数 $\leq 1000$ ）

### 输出

一个整数，表示能够存入背包的最大价值。

### 样例输入

```
5 50
1 1 50
2 4 3
48 49 1
1 51 1
3 3 3
```

### 样例输出

```
106
```

## 5.二维费用背包

二维费用的背包问题是指：对于每件物品，具有两种不同的费用，选择这件物品必须同时付出这两种代价，对于每种代价都有一个客服处的最大值（背包容量）。问：怎样选择物品可以得到最大的价值。设这两种代价分别为代价1和代价2。第i件物品所需的两种代价分别为v[i]和w[i]。

两种代价可付出的最大值（两种背包容量）分别为maxv和maxw，物品的价值为c[i]。

解决办法：费用增加了一维，只需状态也增加一维即可。

设dp[i][j][k]表示前i件物品，背包容量为j，背包承重为k时可以获得的最大值

状态转移方程为：

$$dp[i][j][k] = \max(dp[i-1][j][k], dp[i-1][j-v[i]][k-w[i]] + c[i])$$

空间优化后可以用二维数组求解：

$$dp[j][k] = \max(dp[j][k], dp[j-v[i]][k-w[i]] + c[i])$$

## 5.1 课堂练习

### 1. 最大卡路里

#### 题目描述

神州飞船准备运送一批食品到太空站，该飞船能够运送食品的重量、体积都有严格的限制。

现已知 nn 件完全不同的食品，每种食品的重量、体积及该食品能够提供的卡路里的值，请你编程计算出，该飞船最多能够运送多少卡路里的食物？

### 输入

第一行有两个整数，表示神州飞船能够装载食物的体积最大值(<400<400)和质量最大值(<400<400);

第二行，一个整数 食品总数 NN (<50<50);

第三行 ~~ 第  $3+N_3+N$  行，每行三个数，表示第  $i$  件食品的体积(<400<400) 质量(<400<400) 所含卡路里(<500<500)。

### 输出

一个整，表示所能达到的最大卡路里的值( int 范围内)

### 样例输入

```
320 350
4
160 40 120
80 110 240
220 70 310
40 400 22
```

### 样例输出

```
550
```

## 5.2 课后作业

### 1. 最大购物优惠

#### 题目描述

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。

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
```

## 6.有依赖的背包

### 6.1 课堂案例

#### 1. 采购礼品

##### 题目描述

王老师来到商店为同学们采购礼品。

这家店有  $n$  种礼品（编号是  $1 \sim n$ ），每种礼品只有 11 件。老板为了促销，对礼品进行搭配销售，有关联性的礼品必须都要采购（奇怪的规定），比如 11 号礼品和 33 号礼品搭配了，33 号和 88 号礼品搭配了，那么王老师想要买 11 号礼品的话，就需要把 33 号和 88 号礼品都买了。

现给定每种礼品的价钱和价值，请问在有限的钱  $w$  的情况下，能够买到礼品的最大价值是多少？

##### 输入

第一行输入三个整数， $n, m, w$ ，表示有  $n$  种礼品， $m$  个搭配和你现有的钱的数目。

第二行至  $n+1+n+1$  行，每行有两个整数， $c, d$ ，表示第  $i$  种礼品的价钱和价值。

$(1 \leq c, d \leq 105)$

第  $n+2+n+2$  至  $n+1+mn+1+m$  行，每行有两个整数， $u, v$ ，表示  $u$  号礼品和  $v$  号礼品是有关联的，已经形成搭配销售的关系。

数据范围：

$1 \leq n, w \leq 104, 0 \leq m \leq 5 \times 103$   $1 \leq u, v \leq 104, 0 \leq m \leq 5 \times 103$ 。

##### 输出

一行，表示可以获得的最大价值。

#### 样例输入

```
5 3 10  
3 10  
3 10  
3 10  
5 100  
10 1  
1 3  
3 2  
4 2
```

#### 样例输出

```
1
```

## 2. 金明的预算方案 (有附件的01背包)

### 题目描述

金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间金明自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过  $n$  元钱就行”。今天一早，金明就开始做预算了，他把想买的物品分为两类：主件与附件，附件是从属于某个主件的，下表就是一些主件与附件的例子：

主件	附件
电脑	打印机, 扫描仪
书柜	图书
书桌	台灯, 文具
工作椅	无

如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有 0 个、1 个或 2 个附件。每个附件对应一个主件，附件不再有从属于自己的附件。金明想买的东西很多，肯定会超过妈妈限定的  $n$  元。于是，他把每件物品规定了一个重要度，分为 5 等：用整数 1~5 表示，第 5 等最重要。他还从因特网上查到了每件物品的价格（都是 10 元的整数倍）。他希望在不超过  $n$  元的前提下，使每件物品的价格与重要度的乘积的总和最大。

设第  $j$  件物品的价格为  $v_j$ ，重要度为  $w_j$ ，共选中了  $k$  件物品，编号依次为  $j_1, j_2, \dots, j_k$ ，则所求的总和为：

$$v_{j_1} \times w_{j_1} + v_{j_2} \times w_{j_2} + \dots + v_{j_k} \times w_{j_k}$$

请你帮助金明设计一个满足要求的购物单。

### 输入

第一行有两个整数，分别表示总钱数  $n$  和希望购买的物品个数  $m$ 。

第 22 到第  $(m+1)$  行，每行三个整数，第  $(i+1)$  行的整数  $v_i, p_i, q_i$  分别表示第  $i$  件物品的价格、重要度以及它对应的主件。如果  $q_i=0$ ，表示该物品本身是主件。

### 输出

输出一行一个整数表示答案。

### 样例输入

```
1000 5
800 2 0
400 5 1
300 5 1
400 3 0
500 2 0
```

### 样例输出

```
2200
```

### 说明

### 数据规模与约定

对于全部的测试点，保证  $1 \leq n \leq 3.2 \times 10^4$ ,  $1 \leq m \leq 60$ ,  $0 \leq v_{ij} \leq 10^4$ ,  $1 \leq w_{ij} \leq 5$ ,  $0 \leq q_i \leq m$ , 答案不超过  $2 \times 10^5$ 。

来源：noip2006提高组第2题。

## 7.分组背包

### 1. 分组背包问题

#### 题目描述

有  $N$  组物品和一个容量是  $V$  的背包。每组物品有若干个，同一组内的物品最多只能选一个。

每件物品的体积是  $v_{ij}$ , 价值是  $w_{ij}$ , 其中  $i$  是组号,  $j$  是组内编号。

求解将哪些物品装入背包，可使物品总体积不超过背包容量，且总价值最大。

输出最大价值。

#### 输入

第一行有两个整数  $N, V$ , 用空格隔开, 分别表示物品组数和背包容量。

接下来有  $N$  组数据：

每组数据第一行有一个整数  $S_i$ , 表示第  $i$  个物品组的物品数量。

每组数据接下来有  $S_i$  行, 每行有两个整数  $v_{ij}, w_{ij}$ , 用空格隔开, 分别表示第  $i$  个物品组的第  $j$  个物品的体积和价值。

#### 数据范围

$0 < N, V \leq 100, 0 < S_i \leq 100, 0 < v_{ij}, w_{ij} \leq 100$ 。

#### 输出

输出一个整数, 表示最大价值。

#### 样例输入

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

#### 样例输出

```
8
```

## 8.背包问题问法的变化

### 8.1 01背包求方案数等问题

#### 8.1.1 课堂练习

##### 1. 小明买书

###### 题目描述

新的学习开始了，小明来到书店采购辅导书。

小明有  $M$  元，书店有  $N$  种不同的书，第 $i$  种书卖  $A_i$  元，假设小明每种书最多只能买 1 本，且要花完  $M$  元，请问小明有多少种不同的买书方案？

###### 输入

第一行有两个整数  $N$  和  $M$ 。  
 $(1 \leq N \leq 100, 1 \leq M \leq 10000)$ 。

第二行有  $N$  个整数  $A_i$ 。  
 $(1 \leq A_i \leq 1000)$ 。

###### 输出

输出一个整数，代表买书的方案数。（请注意：本题的计算结果可能会超过int）

###### 样例输入

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

###### 样例输出

```
3
```

##### 2. 数字的组合

###### 题目描述

给定  $N$  个正整数  $A_1, A_2, \dots, A_N$ ，从中选出若干个数，使它们的和为  $M$ ，求有多少种选择方案。

注意：选择不同位置的，但值相同的数，认为是不同的方案。

比如：有 3 个数 1 1 1，要组合出 2，那么有 3 个方案，分别是选第 1、2 个数，选第 1、3 个数，选第 2、3 个数。

###### 输入

第一行包含两个整数  $N$  和  $M$ 。

第二行包含  $N$  个整数，表示  $A_1, A_2, \dots, A_N$ 。

###### 数据范围

$1 \leq N \leq 100, 1 \leq M \leq 10000, 1 \leq A_i \leq 1000$ 。

###### 输出

包含一个整数，表示可选方案数（本题的计算结果一定在int范围内）。

###### 样例输入

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

###### 样例输出

## 8.1.2 课后作业

### 1. 开心的金明

#### 题目描述

金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间他自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过N元钱就行”。今天一早金明就开始做预算，但是他想买的东西太多了，肯定会超过妈妈限定的N元。于是，他把每件物品规定了一个重要度，分为5等：用整数1-5表示，第5等最重要。他还从因特网上查到了每件物品的价格（都是整数元）。他希望在不超过N元（可以等于N元）的前提下，使每件物品的价格与重要度的乘积的总和最大。

设第j件物品的价格为 $v[j]$ ，重要度为 $w[j]$ ，共选中了k件物品，编号依次为 $j_1, j_2, \dots, j_k$ ，则所求的总和为：

$$v[j_1] \times w[j_1] + v[j_2] \times w[j_2] + \dots + v[j_k] \times w[j_k]。$$

请你帮助金明设计一个满足要求的购物单。

#### 输入

第一行，为22个正整数，用一个空格隔开：n,m (其中N(<30000)表示总钱数，m(<25)为希望购买物品的个数。)

从第22行到第m+1行，第j行给出了编号为j-1的物品的基本数据，每行有2个非负整数vp (其中v表示该物品的价格( $v \leq 10000$ )，p表示该物品的重要度(1-5)

#### 输出

1个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值(<100000000)。

#### 样例输入

```
1000 5
800 2
400 5
300 5
400 3
200 2
```

#### 样例输出

```
3900
```

### 2. 背包问题求方案数

#### 题目描述

有 N 件物品和一个容量是 V 的背包。每件物品只能使用一次。

第 i 件物品的体积是  $v_i$ ，价值是  $w_i$ 。

求解将哪些物品装入背包，可使这些物品的总体积不超过背包容量，且总价值最大。

输出 最优选法的方案数。注意答案可能很大，请输出答案模 109+7 的结果。

#### 输入

第一行两个整数， $N, V$ ，用空格隔开，分别表示物品数量和背包容积。

接下来有  $N$  行，每行两个整数  $v_i, w_i$ ，用空格隔开，分别表示第  $i$  件物品的体积和价值。

**数据范围**

$0 < N, V \leq 1000$

$0 < v_i, w_i \leq 1000$

### 输出

输出一个整数，表示方案数模  $10^9 + 7$  的结果。

### 样例输入

```
4 5
```

```
1 2
```

```
2 4
```

```
3 4
```

```
4 6
```

### 样例输出

```
2
```

## 3. 码头的集装箱

### 题目描述

码头上停泊一艘远洋轮船，轮船可以装下  $c$  吨的货物，码头上有  $n$  个集装箱需要运走，已知第  $i$  个集装箱的重量为  $w_i$ 。

请你编程计算，在不超出轮船最大载重量的情况下，该轮船最多可以运走多少吨的集装箱。（注意：单个集装箱不能拆开运送，对于每个集装箱来说，要么整个运到轮船上，要么不运）

### 输入

第一行有 2 个正整数  $n$  和  $c$ 。 $n$  是集装箱数， $c$  是轮船的载重量。

第 2 行中有  $n$  个正整数，表示集装箱的重量 ( $0 < n < 10000, 0 < c < 32767$ )。

### 输出

计算出的最大装载重量输出。

### 样例输入

```
5 10
```

```
7 2 6 5 4
```

### 样例输出

```
10
```

## 4. 装箱问题

### 题目描述

有一个箱子容量为  $V$ （正整数， $0 \leq V \leq 200000$ ），同时有  $n$  个物品（ $0 < n \leq 30$ ），每个物品有一个体积（正整数）。

要求  $n$  个物品中，任取若干个装入箱内，使箱子的剩余空间为最小。

### 输入

1 个整数，表示箱子容量；

1 个整数，表示有 n 个物品；

接下来 n 行，分别表示这 n 个物品的各自体积；

### 输出

1 个整数，表示箱子剩余空间。

### 样例输入

```
24
6
8
3
12
7
9
7
```

### 样例输出

```
0
```

## 5. 集合 Subset Sums

### 题目描述

对于从 1~n 的连续整数集合，能划分成两个子集合，且保证每个集合的数字和是相等的。举个例子，如果 n=3，对于{1,2,3} 能划分成两个子集合，每个子集合的所有数字和是相等的：

{3} 和 {1,2} 是唯一一种分法（交换集合位置被认为是同一种划分方案，因此不会增加划分方案总数）

如果 n=7，有四种方法能划分集合{1,2,3,4,5,6,7}，每一种分法的子集合各数字和是相等的：

{1,6,7} 和 {2,3,4,5}

{2,5,7} 和 {1,3,4,6}

{3,4,7} 和 {1,2,5,6}

{1,2,4,7} 和 {3,5,6}

给出n，你的程序应该输出划分方案总数。

### 输入

输入文件只有一行，且只有一个整数 n

#### 【数据范围】

对于100% 的数据， $1 \leq n \leq 39$ 。

### 输出

输出划分方案总数。

### 样例输入

```
7
```

### 样例输出

```
4
```

## 8.2 完全背包相关问题

### 8.2.1 课堂案例

#### 1. 钱币兑换

##### 题目描述

在一个国家仅有 1 分, 2 分, 3 分硬币, 将钱 N 分 ( $N < 32768$ ) 兑换成硬币有很多种兑法。

请你编程序计算出共有多少种兑法。

##### 输入

输入一个正整数 N, N 小于 32768。

##### 输出

输出兑换的方法数。 (本题数据的计算结果在int范围内)

##### 样例输入

```
2934
```

##### 样例输出

```
718831
```

#### 2. 公交乘车

##### 题目描述

A城市有一条非常特别的街道，该街道在每个公里的节点上都有一个公交车站，乘客可以在任意的公交站点上车，在任意的公交站点下车。乘客根据每次乘坐公交的公里数进行付费，比如，下表就是乘客乘坐不同的公里数要付的费用。 (请注意：不一定公里数越高，费用越高，这也是这条街道特别的地方)

公里数	1	2	3	4	5	6	7	8	9	10
付费金额	12	21	31	40	49	58	69	79	90	101

一辆公交车单次行驶的公里数一定不超过 10 公里，一个乘客如果打算乘坐公交车完成 n 公里 ( $1 \leq n \leq 100$ ) 的行程，他可以选择无限次的换车来完成行程。

请问，他最少要花多少钱？

##### 输入

第一行十个整数分别表示公交行走 1 到 10 公里的费用 ( $\leq 500$ )。注意这些数并无实际的经济意义，即行驶 10 公里费用可能比行驶一公里少。

第二行一个整数 n 表示，旅客的总路程数。 ( $1 \leq n \leq 100$ )

##### 输出

仅一个整数表示最少费用。

##### 样例输入

```
12 21 31 40 49 58 69 79 90 101  
15
```

## 样例输出

147

### 8.2.2 课后作业

#### 1. 货币问题

##### 题目描述

某国家有  $n$  种不同面值的货币，第  $i$  种货币价值  $a_i$  元。

请问：如果每种货币都提供任意多的数量的情况下，如果需要  $m$  元金额的货币，有多少种不同的方案？

##### 输入

第一行两个整数  $n, m$  ( $m \leq 5000, n \leq 100$ )；

以下  $n$  行，每行一个整数，第  $i+1$  行为第  $i$  种货币的面值。

##### 输出

一个整数，为方案数（方案数  $\leq 10^{18}$ ）。

##### 样例输入

```
3 10
1
2
5
```

## 样例输出

10

### 8.3 多重背包相关问题

#### 8.3.1 课堂案例

#### 1. 砝码称重

##### 题目描述

设有 1g、2g、3g、5g、10g、20g 的砝码各若干枚（其总重  $\leq 1000$ ），求这些砝码能称出的不同重量的个数。

##### 输入

读入  $a_1, a_2, a_3, a_4, a_5, a_6$ ，分别表示 1g 砝码有  $a_1$  个，2g 砝码有  $a_2$  个，…… 20g 砝码有  $a_6$  个， $0 \leq a_i \leq 200$ 。

##### 输出

整数  $N$  ( $N$  表示用这些砝码能称出的不同重量的个数，但不包括一个砝码也不用的情况)。

##### 样例输入

```
1 1 0 0 0 0
```

## 样例输出

### 8.3.2 课后作业

#### 1. 奖品采购

##### 题目描述

为了迎接班级的元旦晚会，班主任王老师特批了  $m$  元请你帮忙采购物品，并给你一张清单，注明了王老师调研小卖部中出售的  $n$  个待选物品的价格、价值（价值越高，同学们越喜欢）、以及最多能买到的数量。

请你编程计算出， $m$  元最多能够采购到的物品的最大价值是多少？注意， $m$  元不一定都要花完。

##### 输入

第1行两个数  $n$  ( $n \leq 500$ )， $m$  ( $m < 6000$ )，其中  $n$  代表清单中待选物品总数， $m$  表示王老师的拨款金额。

接下来  $n$  行，每行 3 个数， $p$ 、 $v$ 、 $s$  分别表示第  $i$  种物品的价格、价值和能够买的最大数量（买 0 件到  $s$  件均可），其中  $p \leq 100, v \leq 100, s \leq 10$ 。

##### 输出

输出一个整数，表示能够采购到的最大价值。

##### 样例输入

```
5 1000
80 20 4
40 50 9
30 50 7
40 30 6
20 20 1
```

##### 样例输出

```
1040
```

## 第五章 前缀和、差分

## 第六章 数学知识及编程应用

## 第七章 树及树的应用

## 第八章 二叉树及二叉树的应用

## 第九章 区间DP

# 码蜂C++程序设计——算法提高与进阶

## 第一章 哈希、哈希表与离散化

## 第二章 KMP字符串匹配算法

## 第三章 Trie字典树与AC自动机

## 第四章 树状数组及应用

## 第五章 线段树及应用

## 第六章 倍增、RMO、LCA及树上差分

## 第七章 深搜优化、深搜进阶

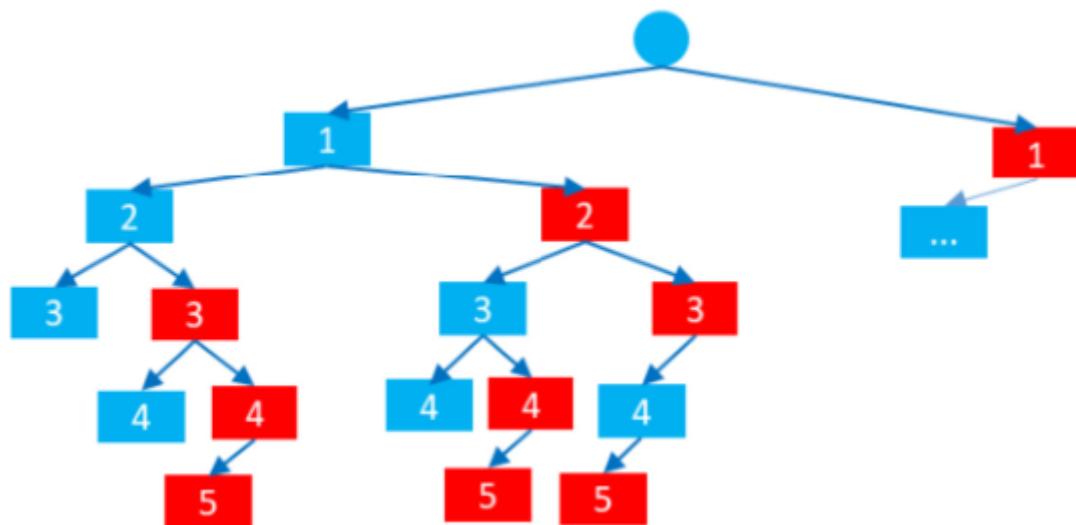
### 1. 深搜剪枝优化

#### 1.1 什么是深搜剪枝优化

搜索算法的时间复杂度大多是指数级的，难以满足对程序时间的限制要求，为使减低时间复杂度，对深度优先搜索可以进行一种优化的基本方法——剪枝。

搜索的进程可以看作是从树的根出发，遍历一整棵树的过程，所谓剪枝，也就是通过某些条件判断避免一些不必要的搜索步骤，因此可以形象地理解为将搜索树的某些枝条“剪掉”，即“剪枝”。

如下图，4数选3个数的组合的搜索树



## 1.2 搜索剪枝技巧

### 1.2.1 优化搜索顺序

在不同的问题中，搜索树的各个层次、各个分支之间的顺序不是固定的，不同的搜索顺序会产生不同的搜索树形态，其规模大小也相差甚远

### 1.2.2 排除等效冗余

在搜索过程中，若能判断从搜索树当前节点上沿某几条不同分支到达的子树是相同的，那么只需要对其中一条分支执行搜索。

### 1.2.3 可行性剪枝/上下界剪枝

可行性剪枝也叫上下界剪枝，是指在搜索过程中及时对当前状态进行检查，若发现分支已无法达到递归边界，就执行回溯。

### 1.2.4 最优化剪枝

在最优化问题的搜索过程中，若当前花费的代价已经超过**当前搜索到的最优解**，那么无论采取多么优秀的策略到达递归边界都不能更新答案，此时可以停止对当前分支的搜索进行回溯。

### 1.2.5 记忆化剪枝

记录每个状态的搜索结果，在重复遍历到同一个状态时，直接返回存储的结果

## 1.3 课堂案例

### 1. 和为K

#### 题目描述

给定一个含N个不同数的数列，任意从数组中选出若干不同的数（也可以选11个），使其和为K，请问有多少种不同的方案。

#### 输入

第1行，输入两个整数N和K。（ $2 \leq N \leq 20, 1 \leq K \leq 10^6$ ）

第2行，有N个用空格隔开的整数。（整数的值在 $[1, 10^6]$ 之间）

#### 输出

输出一个整数，代表方案数。

#### 样例输入

```
5 6
2 1 4 5 3
```

#### 样例输出

```
3
```

**思路：**本题有三种可选优化方案

#### 1.排除等效冗余

采用组合的方式搜索：因为1 2 3 和 1 3 2 是同一种方案

#### 2.搜索顺序优化

从大到小搜索

### 3. 可行性剪枝/上下界剪枝

1. 如果总和>k, 返回
2. 到达搜索边界, 总和 == k, 返回
3. 如果已经选择的总和 + 剩余的所有数的和 <k, 返回
4. 如果讨论的数字边界超过n, 返回

示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int a[30], b[30];
int n, k, ans=0;
/*
讨论到第x个数, 此时的和为sum
*/
void dfs(int x, int sum){

    if(sum==k){
        ans++;
        return;
    }
    if(sum>k) return;
    if(x>n) return;
    //如果已经选中的和加上剩余的数的和<k
    if(sum+b[n]-b[x-1]<k) return;
    dfs(x+1, sum+a[x]);
    dfs(x+1, sum);
}
int main(){
    cin>>n>>k;
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
    sort(a+1, a+n+1);
    reverse(a+1, a+n+1);
    for(int i=1;i<=n;i++){
        b[i]=a[i]+b[i-1];
    }
    dfs(1, 0);
    cout<<ans;
    return 0;
}
```

## 2. 数的划分

### 题目描述

将整数n分成k份, 且每份不能为空, 任意两份不能相同(不考虑顺序)。

例如: n=7, k=3, 下面三种分法被认为是相同的。

{1, 1, 5}; {1, 5, 1}; {5, 1, 1};

问有多少种不同的分法。输出一个整数, 即不同的分法。

## 输入

两个整数n, k( $6 \leq n \leq 200$ ,  $2 \leq k \leq 6$ ), 中间用单个空格隔开。

## 输出

一个整数, 即不同的分法。

## 样例输入

```
7 3
```

## 样例输出

```
4
```

## 说明

四种分法为: {1, 1, 5}; {1, 2, 4}; {1, 3, 3}; {2, 2, 3}。

## 思路:

1. 由于分解不考虑顺序, 因此第*i*份数据点的值 $\leq$ 第*i+1*份数据点的值
2. 假设剩余的未分解的数的和为n, 当前讨论到了第idx份, 则剩余  $(k-idx+1)$ 个数的和要为n, 由于分解不考虑顺序, 因此当前的值不可能超过剩余和的平均值, 也就是当前点的值 $\leq n/(k-idx+1)$

## 示例代码:

```
#include<bits/stdc++.h>
using namespace std;
/*
上下界剪枝:
1. 每份的值 $\geq$ 前一份的值
2. 每份的值 $\leq$ 剩余数的平均数
*/
int n,k;
int a[10];//存储每一份的值
int cnt = 0;
//idx 讨论到了第idx 个数可以填哪些值
void dfs(int idx){
    if(n==0) return;
    if(idx==k){
        //最后一份的值 $\geq$ 前一份的值
        if(n>=a[idx-1]){
            cnt++;
        }
        return;
    }
    //讨论第idx个数的上下界
    for(int i=a[idx-1];i<=n/(k-idx+1);i++){
        a[idx]=i;
        n-=i;//计算剩余数的和
        dfs(idx+1);
        //下标为idx的位置准备填其它值, 恢复n的值
        n+=i;
    }
}
int main(){
    cin>>n>>k;
```

```
a[0]=1;//方便第1个数从1开始讨论  
dfs(1);//从第1份开始讨论  
cout<<cnt;  
return 0;  
}
```

### 3. 单词接龙的最长长度

#### 题目描述

单词接龙是一个与我们经常玩的成语接龙相类似的游戏，现在我们已知一组单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”（每个单词都最多在“龙”中出现两次），在两个单词相连时，其重合部分合为一部分。

例如 beast 和 astonish，如果接成一条龙则变为beastonish，另外相邻的两部分不能存在包含关系，例如 at 和 atide 间不能相连。

#### 输入

输入的第一行为一个单独的整数n ( $n \leq 20$ )表示单词数，以下 n 行每行有一个单词，输入的最后一行为一个单个字符，表示“龙”开头的字母。

你可以假定以此字母开头的“龙”一定存在。

#### 输出

只需输出以此字母开头的最长的“龙”的长度。

#### 样例输入

```
5  
at  
touch  
cheat  
choose  
tact  
a
```

#### 样例输出

```
23
```

#### 说明

##### 样例说明

连成的“龙”为atoucheatactactactouchoose

#### 思路

1. 如果两个单词能接龙，我们要尽可能短的匹配，使得龙更长
2. 为了避免重复的判断两个单词是否能接龙，我们可以预处理一下单词之间的匹配状态，计算出任意两个单词之间最短可以匹配多少个字符，提升搜索的速度

#### 示例代码：

```
#include<bits/stdc++.h>  
using namespace std;  
int n;  
string a[30];  
int d[30][30];//记录第i个单词和第j个单词最短有几个字符能匹配
```

```

int f[30];//标记第i个单词使用的次数
char c;
int ma;//表示龙的最长长度
//讨论到了第idx个单词，此时龙长len
void dfs(int idx,int len){
    ma=max(ma,len);
    f[idx]++;
    //讨论哪个单词能和第idx个单词接龙
    for(int i=1;i<=n;i++){
        if(f[i]<2&&d[idx][i]!=0){
            dfs(i,len+a[i].size()-d[idx][i]);
        }
    }
    f[idx]--;
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++)cin>>a[i];
    cin>>c;
    //计算任意两个单词最短匹配的长度
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            int l1=a[i].size(),l2=a[j].size();
            //枚举可能匹配的长度
            for(int k=1;k<min(l1,l2);k++){
                //如果第i个单词的尾有k个字符和第j个单词的头相同，说明能匹配
                if(a[i].substr(l1-k)==a[j].substr(0,k)){
                    d[i][j]=k;
                    break;
                }
            }
        }
    }
    //从第i个可能的单词开始搜索
    for(int i=1;i<=n;i++){
        if(a[i][0]==c) dfs(i,a[i].size());
    }
    cout<<ma;
    return 0;
}

```

#### 4. 小木棍

##### 题目描述

乔治有一些同样长的小木棍，他把这些木棍随意砍成几段，直到每段的长都不超过50。现在，他想把小木棍拼接成原来的样子，但是却忘记了自己开始时有多少根木棍和它们的长度。给出每段小木棍的长度，编程帮他找出原始木棍的最小可能长度。

##### 输入

第一行为一个单独的整数N表示砍过以后的小木棍的总数，其中 $N \leq 60$ ，第二行为N个用空格隔开的正整数，表示N根小木棍的长度。

##### 输出

仅一行，表示要求的原始木棍的最小可能长度。

##### 样例输入

9

5 2 1 5 2 1 5 2 1

## 样例输出

6

## 思路：

### 剪枝与优化

1. 枚举原始大木棍的可能长度len, len一定在[所有小木棍最大长度, 所有小木棍总长度]的范围内, 该长度一定是所有小木棍和sum的因子
2. 优化搜索顺序: 从大到小枚举, 减少搜索空间
3. 排除等效冗余: 1 2 3, 1 3 2是同一个方案, 因此按组合的方式而非按排列的方式枚举。对于每根大木棍的片段的枚举, 每个片段编号的枚举, 从上一个片段编号idx+1开始枚举, 也就是按照长度降序枚举
4. 排除等效冗余: 如果某小木棍用来拼凑某个长度的木棍失败了, 那么使用长度相等的木棍来拼凑某个长度的木棍也会失败
5. 如果某小木棍作为某大木棍的第一个小木棍拼接失败了, 那么该方案一定失败
6. 如果小木棍作为某大木棍的最后一个小木棍拼接成功, 但是接下来递归失败了那么该方案一定失败

## 示例代码:

```
#include<bits/stdc++.h>
using namespace std;
const int N = 70;
int a[N];
bool f[N]; //标记哪些小木棍被使用过
int n, len, sLen;
//len: 每个枚举的长度
//sLen: 所有小木棍长度和
//将所有的小木棍拼接成若干长度为len的大木棍
//cnt: 已经拼成了cnt个大木棍, sum: 当前正在拼接的大木棍长度
//idx: 当前准备枚举的小木棍的下标值
bool dfs(int cnt, int sum, int idx){
    //如果方案成立
    if(cnt*len==sLen) return true;
    //如果当前大木棍拼完了, 拼下一根
    if(sum==len) return dfs(cnt+1, 0, 1);
    //拼当前这一根大木棍
    int fail=0;
    for(int i=idx; i<=n; i++){
        //如果当前小木棍被用过了, 用这跟小木棍发小大木棍总长度>len
        //当前的小木棍长度之前尝试过是失败的
        if(f[i] || sum+a[i]>len || a[i]==fail) continue;
        f[i]=true;
        //继续搜索
        if(dfs(cnt, sum+a[i], i+1)) return true;
        f[i]=false;
        //当前方案失败, 记录当前的值
        fail=a[i];
    }
}
```

```

        //如果当前小木棍是大木棍的第一个，失败了，那么方案失败，后退
        //如果小木棍作为大木棍的最后一个小木棍拼接成功，后续递归失败了那么该方案一定失败，后退
    if(sum==0||sum+a[i]==len) return false;
}
return false;
}

int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>a[i];
        slen+=a[i];
    }
    sort(a+1,a+1+n);
    reverse(a+1,a+1+n); //优化搜索顺序
    //枚举大木棍长度的可能范围
    for(len=a[1];len<=slen;len++){
        if(slen%len==0&&dfs(0,0,1)){
            cout<<len;
            break;
        }
    }
    return 0;
}

```

## 5. 选数

### 题目描述

已知  $n$  ( $1 \leq n \leq 20$ ) 个整数  $x_1, x_2, \dots, x_n$  ( $1 \leq x_i \leq 5000000$ )，以及一个整数  $k$  ( $k < n$ )。从  $n$  个整数中任选  $k$  个整数相加，可分别得到一系列的和。现在，要求你计算出和为素数共有多少种。

例如当  $n=4$ ,  $k=3$ , 4 个整数分别为 3,7,12,19 时，可得全部的组合与它们的和为：

$3+7+12=22$

$3+7+19=29$

$7+12+19=38$

$3+12+19=34$

现在，要求你计算出和为素数共有多少种。

例如上例，只有一种的和为素数： $3+7+19=29$ 。

### 输入

第一行两个空格隔开的整数  $n, k$  ( $1 \leq n \leq 20$ ,  $k < n$ )。

第二行  $n$  个整数，分别为  $x_1, x_2, \dots, x_n$  ( $1 \leq x_i \leq 5 \times 10^6$ )

### 输出

输出一个整数，表示种类数。

### 样例输入

```

4 3
3 7 12 19

```

### 样例输出

### 思路：

搜索顺序优化：每次搜索当前数的下一个数，选到k个就停止，已经选的数加上剩余所有数< k,也停止。

### 示例代码：

```
#include<bits/stdc++.h>
using namespace std;
int n,k,a[30];
int ans = 0;
bool prime(int x){
    if(x<=1) return false;
    for(int i=2;i<=sqrt(x);i++){
        if(x%i==0) return false;
    }
    return true;
}
//讨论到第x个数，个数为cnt，和为sum
void dfs(int x,int cnt,int sum){
    //剪掉无效递归
    if(cnt==k){
        if(prime(sum)) ans++;
        return;
    }
    if(x>n) return;
    if(cnt+n-x+1<k) return;
    //选择第x个数
    dfs(x+1,cnt+1,sum+a[x]);
    //不选第x个数
    dfs(x+1,cnt,sum);
}
int main(){
    cin>>n>>k;
    for(int i=1;i<=n;i++) cin>>a[i];
    //从第1个数开始讨论选或者不选
    //选中的数的个数是0，总和是0
    dfs(1,0,0);
    cout<<ans;
    return 0;
}
```

**第八章 广搜优化、广搜进阶**

---

**第九章 迭代加深搜索、启发式搜索、A\*、IDA\***

---

**第十章 图论进阶**

---

**第十一章 图论进阶(2)**

---

**第十二章 单调栈和单调队列**

---

**第十三章 树形DP**

---

**第十四章 数位DP**

---

**第十五章 压状DP、DP的单调队列优化**

---

**第十六章 平衡树**

---

**第十七章 编程中的数学**

---

**第十八章 排序算法**

---