51.谁是窃贼

公安人员审问四名窃贼嫌疑犯。已知，这四人当中仅有一名是窃贼，还知道这四人中每人要么是诚实的，要么总是说谎的。在回答公安人员的问题中：  
甲说：“乙没有偷，是丁偷的。”  
乙说：“我没有偷，是丙便的。”  
丙说：“甲没有偷，是乙偷的。”  
丁说：“我没有偷。”  
请根据这四人的答话判断谁是盗窃者。

**\*问题分析与算法设计**  
假设A、B、C、D分别代表四个人，变量的值为1代表该人是窃贼。  
由题目已知：四人中仅有一名是窃贼，且这四个人中的每个人要么说真话，要么说假话，而由于甲、乙、丙三人都说了两句话：“X没偷，X偷了”，故不论该人是否说谎，他提到的两人中必有一人是小偷。故在列条件表达式时，可以不关心谁说谎，谁说实话。这样，可以列出下列条件表达式：  
甲说：”乙没有偷，是丁偷的。” B+D=1  
乙说：“我没有偷，是丙偷有。” B+C=1  
丙说：“甲没有偷，是乙偷的。” A+B=1  
丁说：“我没有偷。” A+B+C+D=1  
其中丁只说了一句话，无法判定其真假，表达式反映了四人中仅有一名是窃贼的条件。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int i,j,a[4];  
for(i=0;i<4;i++) /\*假定只有第i个人为窃贼\*/  
{  
for(j=0;j<4;j++) /\*将第i个人设置为1表示窃贼，其余为0\*/  
if(j==i)a[j]=1;  
else a[j]=0;  
if(a[3]+a[1]==1&&a[1]+a[2]==1&&a[0]+a[1]==1) /\*判断条件是否成立\*/  
{  
printf("The thief is "); /\*成立\*/  
for(j=0;j<=3;j++) /\*输出计算结果\*/  
if(a[j])printf("%c.",j+'A');  
printf("/n");  
}  
}  
}

**\*运行结果**  
The thief is B. (乙为窃贼。)

 52.黑与白

有A、B、C、D、E五人，每人额头上都帖了一张黑或白的纸。五人对坐，每人都可以看到其它人额头上的纸的颜色。五人相互观察后，  
A说：“我看见有三人额头上帖的是白纸，一人额头上帖的是黑纸。”  
B说：“我看见其它四人额头上帖的都是黑纸。”  
C说：“我看见一人额头上帖的是白纸，其它三人额头上帖的是黑纸。”  
D说：“我看见四人额头上帖的都是白纸。”  
E什么也没说。  
现在已知额头上帖黑纸的人说的都是谎话，额头帖白纸的人说的都是实话。问这五人谁的额头是帖白纸，谁的额头是帖黑纸？

**\*问题分析与算法设计**  
假如变量A、B、C、D、E表示每个人额头上所帖纸的颜色，0 代表是黑色，1 代表是白色。根据题目中A、B、C、D四人所说的话可以总结出下列关系：  
A说： a&&b+c+d+e==3||!a&&b+c+d+e!=3  
B说： b&&a+c+d+e==0||!b&&a+c+d+e!=0  
C说： c&&a+b+d+e==1||!c&&a+b+d+e!=1  
D说： d&&a+b+c+e==4||!d&&a+b+c+e!=4  
穷举每个人额头所帖纸的颜色的所有可能的情况，代入上述表达式中进行推理运算，使上述表达式为“真”的情况就是正确的结果。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int a,b,c,d,e;  
for(a=0;a<=1;a++) /\*黑色：0 白色：1\*/  
for(b=0;b<=1;b++) /\*穷举五个人额头帖纸的全部可能\*/  
for(c=0;c<=1;c++)  
for(d=0;d<=1;d++)  
for(e=0;e<=1;e++)  
if((a&&b+c+d+e==3||!a&&b+c+d+e!=3)  
&&(b&&a+c+d+e==0||!b&&a+c+d+e!=0)  
&&(c&&a+b+d+e==1||!c&&a+b+d+e!=1)  
&&(d&&a+b+c+e==4||!d&&a+b+c+e!=4))  
{  
printf("A is pasted a piece of %s paper on his forehead./n",  
a?"white":"black");  
printf("B is pasted a piece of %s paper on his forehead./n",  
b?"white":"black");  
printf("C is pasted a piece of %s paper on his forehead./n",  
c?"white":"black");  
printf("D is pasted a piece of %s paper on his forehead./n",  
d?"white":"black");  
printf("E is pasted a piece of %s paper on his forehead./n",  
e?"white":"black");  
}  
}

**\*运行结果**  
A is pasted a paper of black paper on his forehead. (黑)  
B is pasted a paper of black paper on his forehead. (黑)  
C is pasted a paper of white paper on his forehead. (白)  
D is pasted a paper of black paper on his forehead. (黑)  
E is pasted a paper of white paper on his forehead. (白)

53.迷语博士的难题(1)

诚实族和说谎族是来自两个荒岛的不同民族，诚实族的人永远说真话，而说谎族的人永远说假话。迷语博士是个聪明的人，他要来判断所遇到的人是来自哪个民族的。  
迷语博士遇到三个人，知道他们可能是来自诚实族或说谎族的。为了调查这三个人是什么族的，博士分别问了他们的问题，这是他们的对话：  
问第一个人：“你们是什么族？”，答：“我们之中有两个来自诚实族。”第二个人说：“不要胡说，我们三个人中只有一个是诚实族的。”第三个人听了第二个人的话后说：“对，就是只有一个诚实族的。”  
请根据他的回答判断他们分别是哪个族的。

**\*问题分析与算法设计**  
假设这三个人分别为A、B、C，若说谎其值为0，若诚实，其值为1。根据题目中三个人的话可分别列出：  
第一个人： a&&a+b+c==2||!a&&a+b+c!=2  
第二个人： b&&a+b+c==1||!b&&a+b+c!=1  
第三个人： c&&a+b+c==1||!c&&a+b+c!=1  
利用穷举法，可以很容易地推出结果。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int a,b,c;  
for(a=0;a<=1;a++) /\*穷举每个人是说谎还是诚实的全部情况\*/  
for(b=0;b<=1;b++) /\*说谎：0 诚实：1\*/  
for(c=0;c<=1;c++)  
if((a&&a+b+c==2||!a&&a+b+c!=2) /\*判断是否满足题意\*/  
&&(b&&a+b+c==1||!b&&a+b+c!=1)  
&&(c&&a+b+c==1||!c&&a+b+c!=1))  
{  
printf("A is a %s./n",a?"honest":"lier"); /\*输出判断结果\*/  
printf("B is a %s./n",b?"honest":"lier");  
printf("C is a %s./n",c?"honest":"lier");  
}  
}

**\*运行结果**  
A is a lier (说谎族)  
B is a lier (说谎族)  
C is a lier (说谎族)

**\*思考题**  
迷语博士遇到四个人，知道他们可能是来自诚实族和说谎族的。为了调查这四个人是什么族的，博士照例进行询问：”你们是什么族的？“  
第一人说：”我们四人全都是说谎族的。“  
第二人说：”我们之中只有一人是说谎族的。“  
第三人说：”我们四人中有两个是说谎族的。“  
第四人说：”我是诚实族的。“  
问自称是“诚实族”的第四个人是否真是诚实族的？  
(答案：第四个人是诚实族的。)

54.迷语博士的难题(2)

两面族是荒岛上的一个新民族，他们的特点是说话真一句假一句且真假交替。如果第一句为真，则第二句是假的；如果第一句为假的，则第二句就是真的，但是第一句是真是假没有规律。  
迷语博士遇到三个人，知道他们分别来自三个不同的民族：诚实族、说谎族和两面族。三人并肩站在博士前面。  
博士问左边的人：“中间的人是什么族的？”，左边的人回答：“诚实族的”。  
博士问中间的人：“你是什么族的？”，中间的人回答：“两面族的”。  
博士问右边的人：“中间的人究竟是什么族的？”，右边的人回答：“说谎族的”。  
请问：这三个人都是哪个民族的？

**\*问题分析与算法设计**  
这个问题是两面族问题中最基本的问题，它比前面只有诚实族和说谎族的问题要复杂。解题时要使用变量将这三个民族分别表示出来。  
令：变量A=1表示：左边的人是诚实族的(用C语言表示为A)；  
变量B=1表示：中间的人是诚实族的(用C语言表示为B)；  
变量C=1表示：右边的人是诚实族的(用C语言表示为C)；  
变量AA=1表示：左边的人是两面族的(用C语言表示为AA)；  
变量BB=1表示：中间的人是两面族的(用C语言表示为BB)；  
变量CC=1表示：右边的人是两面族的(用C语言表示为CC)；  
则左边的人是说谎族可以表示为：A!=1且AA!=1 (不是诚实族和两面族的人)  
用C语言表示为：!A&&!AA  
中间的人是说谎族可以表示为：B!=1且BB!=1  
用C语言表示为：!B&&!BB  
右边的人是说谎族可以表示为：C!=0且CC!=1  
用C语言表示为：!C&&!CC  
根据题目中“三人来自三个民族”的条件，可以列出：  
a+aa!=2&&b+bb!=2&&c+cc!=2 且 a+b+c==1&&aa+bb+cc==1  
根据左边人的回答可以推出：若他们是诚实族，则中间的人也是诚实族；若他不是诚实族，则中间的人也不是诚实族。以上条件可以表示为：  
c&&!b&&!bb||(!c&&!cc)&&(b||bb)||!c&&cc  
将全部逻辑条件联合在一起，利用穷举的方法求解，凡是使上述条件同时成立的变量取值就是题目的答案。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int a,b,c,aa,bb,cc;  
for(a=0;a<=1;a++) /\*穷举全部情况\*/  
for(b=0;b<=1;b++)  
for(c=0;c<=1;c++)  
for(aa=0;aa<=1;aa++)  
for(bb=0;bb<=1;bb++)  
for(cc=0;cc<=1;cc++)  
if(a+aa!=2&&b+bb!=2&&c+cc!=2&& /\*判断逻辑条件\*/  
a+b+c==1&&aa+bb+cc==1 &&  
(a&&!aa&&b&&!bb||!a&&!b)&&  
!b &&  
(c&&!b&&!bb||(!c&&!cc)&&(b||bb)||!c&cc))  
{  
printf("The man stand on left is a %s./n",  
aa?"double–dealer":(a?"honest":"lier"));  
printf("The man stand on left is a %s./n",  
bb?"double–dealer":(b?"honest":"lier"));  
printf("The man stand on left is a %s./n",  
cc?"double–dealer":(c?"honest":"lier"));   
/\*输出最终的推理结果\*/  
}  
}

**\*运行结果**  
The man stand on left is a double–dealer. (左边的人是两面族的)  
The man stand on center is a lier. (中间的人是说谎族的)  
The man stand on right is a honest. (右边的人是诚实族的)

**\*思考题**  
迷语博士遇到三个人，便问第一个人：“你是什么族的？”，回答：“诚实族的。”问第二个人：“你是什么族的？”，答：“说谎族的。”博士又问第二个人：“第一个人真的是诚实族的吗？”，答：“是的。”问第三个人：“你是什么族的？”，答：“诚实族的。”博士又问第三个人：“第一个人是什么族的？”，答：“两面族的。”  
请判断这个人到底是哪个民族的？  
(答案：第一个人是诚实族的，第二个人是两面族的，第三人是说谎族。)

55.哪个大夫哪天值班

医院有A、B、C、D、E、F、G七位大夫，在一星期内(星期一至星期天)每人要轮流值班一天。现在已知：  
A大夫比C大夫晚一天值班；  
D大夫比E大夫晚二天值班；  
B大夫比G大夫早三天值班；  
F大夫的值班日在B和C大夫的中间，且是星期四；  
请确定每天究竟是哪位大夫值班？

**\*问题分析与算法设计**  
由题目可推出如下已知条件：  
\*F是星期四值班；  
\*B值班的日期在星期一至星期三，且三天后是G值班；  
\*C值班的日期在星期五至星期六，且一天后是A值班；  
\*E两天后是D值班；E值班的日期只能在星期一至星期三；  
在编程时用数组元素的下标1到7表示星期一到星期天，用数组元素的值分别表示A~F七位大夫。

**\*程序说明与注释**  
#include<stdio.h>  
#include<stdlib.h>  
int a[8];  
char \*day[]={"","MONDAY","TUESDAY","WEDNESDAY","THURSDAYT",  
"FRIDAY","SATUDAY","SUNDAY"}; /\*建 立星期表\*/  
int main()  
{  
int i,j,t;  
a[4]=6; /\*星期四是F值班\*/   
for(i=1;i<=3;i++)  
{  
a[i]=2; /\*假设B值班的日期\*/  
if(!a[i+3]) a[i+3]=7; /\*若三天后无人值班则安排G值班\*/  
else{ a[i]=0;continue;} /\*否则B值班的日期不断对\*/  
for(t=1;t<=3;t++) /\*假设E值班的时间\*/  
{  
if(!a[t]) a[t]=5; /\*若当天无人值班则安排E值班\*/  
else continue;  
if(!a[t+2]) a[t+2]=4; /\*若E值班两天后无人值班则应为D\*/  
else{ a[t]=0;continue;} /\*否则E值班的日期不对\*/  
for(j=5;j<7;j++)  
{  
if(!a[j]) a[j]=3; /\*若当天无人值班，则安排C值班\*/  
else continue;  
if(!a[j+1]) a[j+1]=1; /\*C之后一天无人值班则应当是A值班\*/  
else{ a[j]=0;continue;} /\*否则A值班日期不对\*/  
for(i=1;i<=7;i++) /\*安排完毕，输出结果\*/  
printf("Doctor %c is on duty %s./n",'A'+a[i]-1,day[i]);  
exit(0);  
}  
}  
}  
}

**\*运行结果**  
Doctor E is on duty MONDAY. (星期一：E)  
Doctor B is on duty TUESDAY. (星期二：B)  
Doctor D is on duty WEDNESDAY. (星期三：D)  
Doctor F is on duty THUESDAY. (星期四：F)  
Doctor G is on duty FRIDAY. (星期五：G)  
Doctor C is on duty SATURDAY. (星期六：C)  
Doctor A is on duty SUNDAY. (星期日：A)

**\*思考题**  
在本题的求解过程中，我们只考虑了一星期之内的情况，没有考虑跨周的情况。对于“B大夫比G大夫早三天值班的”条件只是简单的认为是在同一周内早三天。若考虑跨周的情况就可能出现：B大夫星期一值班，而G大夫是上周的星期五。同样，对“F大夫的值班日在B和C大夫的中间”这个条件，也可以扩展为：“只要F大夫的值班日在B和C大夫的中间就可以”。  
请考虑允许跨周的情况下，可能的时间安排表。

56.区分旅客国籍

在一个旅馆中住着六个不同国籍的人，他们分别来自美国、德国、英国、法国、俄罗斯和意大利。他们的名字叫A、B、C、D、E和F。名字的顺序与上面的国籍不一定是相互对应的。现在已知：  
1)A美国人是医生。  
2)E和俄罗斯人是技师。  
3)C和德国人是技师。  
4)B和F曾经当过兵，而德国人从未参过军。  
5)法国人比A年龄大；意大利人比C年龄大。  
6)B同美国人下周要去西安旅行，而C同法国人下周要去杭州度假。  
试问由上述已知条件，A、B、C、D、E和F各是哪国人？

**\*问题分析与算法设计**  
首先进行题目分析，尽可能利用已知条件，确定谁不是哪国人。  
由：1) 2) 3)可知：A不是美国人，E不是俄罗斯人，C不是德国人。另外因为A与德国人的职业不同，E与美、德人的职业不同，C与美、俄人的职业不同，故A不是俄罗斯人或德国人，E不是美国人或德国人，C不是美国人或俄罗斯人。  
由4)和5)可知B和F不是德国人，A不是法国人，C不是意大利人。  
由6)可知B不是美国人，也不是法国人(因B与法国人下周的旅行地点不同)；C不是法国人。  
将以上结果汇总可以得到下列条件矩阵：  
. 美(医生) 英 法 德(技师) 意大利 俄(教师)   
A(医生) X . X X . X   
B X . X X . .   
C(技师) X . X X X X   
D . . . . . .   
E(教师) X . . X . X   
F . . . X . .

根据此表使用消元法进行求解，可以方便地得到问题的答案。  
将条件矩阵输入计算机，用程序实现消去算法是很容易的。

**\*程序说明与注释**  
#include<stdio.h>  
char \*m[7]={" ","U.S","U.K","FRANCE","GER","ITALI","EUSSIAN"}; /\*国名\*/  
int main()  
{  
int a[7][7],i,j,t,e,x,y;  
for(i=0;i<7;i++) /\*初始化条件矩阵\*/  
for(j=0;j<7;j++) /\*行为人，列为国家，元素的值表示某人是该国人\*/  
a[i][j]=j;  
for(i=1;i<7;i++) /\*条件矩阵每一列的第0号元素作为该列数据处理的标记\*/  
a[0][i]=1; /\*标记该列尚未处理\*/  
a[1][1]=a[2][1]=a[3][1]=a[5][1]=0; /\*输入条件矩阵中的各种条件\*/  
a[1][3]=a[2][3]=a[3][3]=0; /\*0表示不是该国的人\*/  
a[1][4]=a[2][4]=a[3][4]=a[5][4]=a[6][4]=0;  
a[3][5]=0;  
a[1][6]=a[3][6]=a[5][6]=0;  
while(a[0][1]+a[0][2]+a[0][3]+a[0][4]+a[0][5]+a[0][6]>0)  
{ /\*当所有六列均处理完毕后退出循环\*/  
for(i=1;i<7;i++) /\*i:列坐标\*/  
if(a[0][i]) /\*若该列尚未处理，则进行处理\*/  
{  
for(e=0,j=1;j<7;j++) /\*j:行坐标 e:该列中非0元素计数器\*/  
if(a[j][i]) { x=j;y=i;e++;}  
if(e==1) /\*若该列只有一个元素为非零，则进行消去操作\*/  
{  
for(t=1;t<7;t++)  
if(t!=i)a[x][t]=0; /\*将非零元素所在的行的其它元素置0\*/  
a[0][y]=0; /\*设置该列已处理完毕的标记\*/  
}  
}  
}  
for(i=1;i<7;i++) /\*输出推理结果\*/  
{  
printf("%c is coming from ",'A'-1+i);  
for(j=1;j<7;j++)  
if(a[i][j]!=0)  
{ printf("%s./n",m[a[i][j>); break;}  
}  
}

**\*运行结果**  
A is coming from ITALY. (意大利人)  
B is coming from EUSSIAN. (俄罗斯人)  
C is coming from U.K.. (英国人)  
D is coming from GER. (德国人)  
E is coming from FRANCE. (法国人)  
F is coming from U.S.. (美国人)

**\*问题的进一步讨论**  
生成条件矩阵然后使用消去法进行推理判断是一种常用的方法。对于解决较为复杂的逻辑问题是十分有效的。

**\*思考题**  
地理课上老师给出一张没有说明省份的中国地图，从中选出五个省从1到5编号，要大家写出省份的名称。交卷后五位同学每人只答了二个省份的名称如下，且每人只答对了一个省，问正确答案是什么？  
A 答：2号陕西，5号甘肃 B 答：2号湖北，4号山东  
C 答：1号山东，5号吉林 D 答：3号湖北，4号吉林  
E 答：2号甘肃，3号陕西

57.谁家孩子跑最慢

张王李三家各有三个小孩。一天，三家的九个孩子在一起比赛短跑，规定不分年龄大小，跑第一得9分，跑第2得8分，依此类推。比赛结果各家的总分相同，且这些孩子没有同时到达终点的，也没有一家的两个或三个孩子获得相连的名次。已知获第一名的是李家的孩子，获得第二的是王家的孩子。问获得最后一名的是谁家的孩子？

**\*问题分析与算法设计**  
按题目的条件，共有1+2+3+…+9=45分，每家的孩子的得分应为15分。根据题意可知：获第一名的是李家的孩子，获第二名的是王家的孩子，则可推出：获第三名的一定是张家的孩子。由“这些孩子没有同时到达终点的”可知：名次不能并列，由“没有一家的两个或三个孩子获得相连的名次”可知：第四名不能是张家的孩子。  
程序中为了方便起见，直接用分数表示。

**\*程序说明与注释**  
#include<stdio.h>  
int score[4][4];  
int main()  
{  
int i,j,k,who;  
score[1][1]=7; /\*按已知条件进行初始化：score[1]:张家三个孩子的得分\*/   
score[2][1]=8; /\*score[2]:王家三个孩子的得分\*/  
score[3][1]=9; /\*李家三个孩子的得分\*/  
for(i=4;i<6;i++) /\*i:张家孩子在4到6分段可能的分数\*/  
for(j=4;j<7;j++) /\*j:王家孩子在4到6分段可能的分数\*/  
for(k=4;i!=j&&k<7;k++) /\*k:李家孩子在4到6分段可能的分数\*/  
if(k!=i&&k!=j&&15-i-score[1][1]!=15-j-score[2][1] /\*分数不能并列\*/  
&&15-i-score[1][1]!=15-k-score[3][1]  
&&15-j-score[2][1]!=15-k-score[3][1])  
{  
score[1][2]=i;score[1][3]=15-i-7; /\*将满足条件的结果记入数组\*/  
score[2][2]=j;score[2][3]=15-j-8;  
score[3][2]=k;score[3][3]=15-k-9;  
}  
for(who=0,i=1;i<=3;i++,printf("/n"))  
for(j=1;j<=3;j++)  
{  
printf("%d",score[i][j]); /\*输出各家孩子的得分情况\*/  
if(score[i][j]==1)who=i; /\*记录最后一名的家庭序号\*/  
}  
if(who==1) /\*输出最后判断的结果\*/  
printf("The last one arrived to end is a child from family Zhang./n");  
else if(who==2)  
printf("The last one arrived to end is a child from family Wang./n");  
else printf("The last one arrived to end is a child from family Li./n");  
}

**\*运行结果**  
7 5 3  
8 6 1  
9 4 2  
The last one arrived to end is a child from family Wang.  
(获得最后一名的是王家的孩子。

58.拉丁方阵

构造 NXN 阶的拉丁方阵(2<=N<=9)，使方阵中的每一行和每一列中数字1到N只出现一次。如N=4时：  
1 2 3 4  
2 3 4 1  
3 4 1 2  
4 1 2 3

**\*问题分析与算法设计**  
构造拉丁方阵的方法很多，这里给出最简单的一种方法。观察给出的例子，可以发现：若将每 一行中第一列的数字和最后一列的数字连起来构成一个环，则该环正好是由1到N顺序构成；对于第i行，这个环的开始数字为i。按照 此规律可以很容易的写出程序。下面给出构造6阶拉丁方阵的程序。

**\*程序说明与注释**  
#include<stdio.h>  
#define N 6 /\*确定N值\*/  
int main()  
{  
int i,j,k,t;  
printf("The possble Latin Squares of order %d are:/n",N);  
for(j=0;j<N;j++) /\*构造N个不同的拉丁方阵\*/  
{  
for(i=0;i<N;i++)  
{  
t=(i+j)%N; /\*确定该拉丁方阵第i 行的第一个元素的值\*/  
for(k=0;k<N;k++) /\*按照环的形式输出该行中的各个元素\*/  
printf("%d",(k+t)%N+1);  
printf("/n");  
}  
printf("/n");  
}  
}

**\*运行结果**  
The possble Latin Squares of order 6 are:  
1 2 3 4 5 6 2 3 4 5 6 1 3 4 5 6 1 2  
2 3 4 5 6 1 3 4 5 6 1 2 4 5 6 1 2 3  
3 4 5 6 1 2 4 5 6 1 2 3 5 6 1 2 3 4  
4 5 6 1 2 3 5 6 1 2 3 4 6 1 2 3 4 5  
5 6 1 2 3 4 6 1 2 3 4 5 1 2 3 4 5 6  
6 1 2 3 4 5 1 2 3 4 5 6 2 3 4 5 6 1

4 5 6 1 2 3 5 6 1 2 3 4 6 1 2 3 4 5  
5 6 1 2 3 4 6 1 2 3 4 5 1 2 3 4 5 6  
6 1 2 3 4 5 1 2 3 4 5 6 2 3 4 5 6 1  
1 2 3 4 5 6 2 3 4 5 6 1 3 4 5 6 1 2  
2 3 4 5 6 1 3 4 5 6 1 2 4 5 6 1 2 3  
3 4 5 6 1 2 4 5 6 1 2 3 5 6 1 2 3 4

59.填表格

将1、2、3、4、5和6 填入下表中，要使得每一列右边的数字比左边的数字大，每一行下面的数字比上面的数字大。按此要求，可有几种填写方法？

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

**\*问题分析与算法设计**  
按题目的要求进行分析，数字1一定是放在第一行第一列的格中，数字6一定是放在第二行第三列的格中。在实现时可用一个一维数组表示，前三个元素表示第一行，后三个元素表示第二行。先根据原题初始化数组，再根据题目中填 写数字的要求进行试探。

**\*程序说明与注释**  
#include<stdio.h>  
int jud1(int s[]);  
void print(int u[]);  
int count; /\*计数器\*/  
int main()  
{  
static int a[]={1,2,3,4,5,6}; /\*初始化数组\*/  
printf("The possble table satisfied above conditions are:/n");  
for(a[1]=a[0]+1;a[1]<=5;++a[1]) /\*a[1]必须大于a[0]\*/  
for(a[2]=a[1]+1;a[2]<=5;++a[2]) /\*a[2]必须大于a[1]\*/  
for(a[3]=a[0]+1;a[3]<=5;++a[3]) /\*第二行的a[3]必须大于a[0]\*/  
for(a[4]=a[1]>a[3]?a[1]+1:a[3]+1;a[4]<=5;++a[4])  
/\*第二行的a[4]必须大于左侧a[3]和上边a[1]\*/  
if(jud1(a)) print(a); /\*如果满足题意，打印结果\*/  
}

int jud1(int s[])  
{  
int i,l;  
for(l=1;l<4;l++)  
for(i=l+1;i<5;++i)  
if(s[l]==s[i]) return 0; /\*若数组中的数字有重复的，返回0\*/  
return 1; /\*若数组中的数字没有重复的，返回1\*/  
}

void print(int u[])  
{  
int k;  
printf("/nNo.:%d",++count);  
for(k=0;k<6;k++)  
if(k%3==0) /\*输出数组的前三个元素作为第一行\*/  
printf("/n%d",u[k]);  
else /\*输出数组的后三个元素作为第二行\*/  
printf("%d",u[k]);  
}

**\*运行结果**  
The possble table satisfied above conditions are:  
No.1: No.2: No.3: No.4: No.5:  
1 2 3 1 2 4 1 2 5 1 3 4 1 3 5  
4 5 6 3 5 6 3 4 6 2 5 6 2 4 6

60.1~9分成1：2：3的三个3位数

将1到9 这九个数字分成三个3位数，分求第一个3位数，正好是第二个3位数的二倍，是第三个3位数的三倍。问应当怎样分法。

**\*问题分析与算法设计**  
问题中的三个数之间是有数学关系的，实际上只要确定第一个三位数就可以解决问题。  
试探第一个三位数之后，计算出另外两个数，将其分别分解成三位数字，进行判断后确定所试探的数是否就是答案。  
需要提醒的是：试探的初值可以是123，最大值是333。因为不可能超出该范围。  
\*程序与程序设计  
#include<stdio.h>  
int ok(int t,int \*z);  
int a[9];  
int main()  
{  
int m,count=0;  
for(m=123;m<=333;m++) /\*试探可能的三位数\*/  
if(ok(m,a)&&ok(2\*m,a+3)&&ok(3\*m,a+6)) /\*若满足题意\*/  
printf("No.%d: %d %d %d/n",++count,m,2\*m,3\*m); /\*输出结果\*/  
}

int ok(int t,int \*z) /\*分解t的值，将其存入z指向的三个数组元素，若满足要求返回1\*/  
{  
int \*p1,\*p2;  
for(p1=z;p1<z+3;p1++)  
{  
\*p1=t%10; /\*分解整数\*/  
t/=10;  
for(p2=a;p2<p1;p2++) /\*查询分解出的数字是否已经出现过\*/  
if(\*p1==0||\*p2==\*p1)return 0; /\*若重复则返回\*/  
}  
return 1; /\*否则返回1\*/  
}

**\*运行结果**  
No.1:192 384 576  
No.2:219 438 657  
No.3:273 546 819  
No.4:327 654 981

**\*思考题**  
求出所有可能的以下形式的算式，每个算式中有九个数位，正好用尽1到9这九个数字。  
1)○○○+○○○=○○○ (共有168种可能的组合)  
2)○×○○○○=○○○○ (共有2种可能的组合)  
3)○○×○○○=○○○○ (共有7种可能的组合)  
4)○×○○○=○○×○○○ (共有13种可能的组合)  
5)○×○○○=○×○○○○ (共有28种可能的组合)  
6)○○×○○=○×○○○○ (共有7种可能的组合)  
7)○○×○○=○○×○○○ (共有11种可能的组合)

61.1~9组成三个3位的平方数

将1、2、3、4、5、6、7、8、9九个数字分成三组，每个数字只能用一次，即每组三个数不允许有重复数字，也不许同其它组的三个数字重复，要求每组中的三位数都组成一个平方数。

**\*问题分析与算法设计**  
本问题的思路很多，这里介绍一种简单快速的算法。  
首先求出三位数中不包含0且是某个整数平方的三位数，这样的三位数是不多的。然后将满足条件的三位数进行组合，使得所选出的3个三位数的9个数字没有重复。  
程序中可以将寻找足条件的三位数的过程和对该三位数进行数字分解的过程结合起来。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int a[20],num[20][3],b[10]; /\*a:存放满足条件的三位数\*/  
/\*若不是10 的倍数，则分解三位数\*/  
/\*分解该三位数中的每一个数字\*/  
int i,j,k,m,n,t,flag;  
printf("The 3 squares with 3 different digits each are:/n");  
for(j=0,i=11;i<=31;i++) /\*求出是平方数的三位数\*/  
if(i%10!=0) /\*若不是10的倍数，则分解三位数\*/  
{  
k=i\*i; /\*分解该三位数中的每一个数字\*/  
num[j+1][0]=k/100; /\*百位\*/  
num[j+1][1]=k/10%10; /\*十位\*/  
num[j+1][2]=k%10; /\*个位\*/  
if(!(num[j+1][0]==num[j+1][1]||num[j+1][0]==num[j+1][2]||  
num[j+1][1]==num[j+1][2])) /\*若分解的三位数字均不相等\*/  
a[++j]=k; /\*j:计数器，统计已找到的满足要求的三位数\*/  
}  
for(i=1;i<=j-2;++i) /\*从满足条件的三位数中选出三个进行组合\*/  
{  
b[1]=num[i][0];  
b[2]=num[i][1];  
b[3]=num[i][2];  
for(t=i+1;t<=j-1;++t)  
{  
b[4]=num[t][0]; /\*取第t个数的三位数字\*/  
b[5]=num[t][1];  
b[6]=num[t][2];  
for(flag=0,m=1;!flag&&m<=3;m++) /\*flag:出现数字重复的标记\*/  
for(n=4;!flag&&n<=6;n++) /\*判断两个数的数字是否有重复\*/  
if(b[m]==b[n])flag=1; /\*flag=1:数字有重复\*/  
if(!flag)  
for(k=t+1;k<=j;k++)  
{  
b[7]=num[k][0]; /\*取第k个数的三位数字\*/  
b[8]=num[k][1];  
b[9]=num[k][2];  
for(flag=0,m=1;!flag&&m<=6;m++) /\*判断前两个数字是否\*/  
for(n=7;!flag&&n<=9;n++) /\*与第三个数的数字重复\*/  
if(b[m]==b[n])flag=1;  
if(!flag) /\*若均不重复则打印结果\*/  
printf("%d,%d,%d/n",a[i],a[t],a[k]);  
}  
}  
}  
}

**\*运行结果**  
The 3 squares with 3 different digits each are:  
361,529,784

**\*思考题**  
将1、2、3、4、5、6、7、8、9九个数字分成二组，每个数字只能用一次，一组形成一个5位数，另一组形成一个4位数，使得前者为后者的n倍。求所有满足条件的5位数和4位数。(注意：N的最大值等于68,68以内的某些N也是不可能的。不可能的N值包括：1、10、11、20、21、25、30、31等共32个。)

62.由8个整数形成奇特的立方体

任意给出8个整数，将这8个整数分别放在一个立方体的八个顶点上，要求每个面上的四个数之和相等。

**\*问题分析与算法设计**  
简化问题：将8个顶点对应数组中的8个元素，将“每个面上的四个数之和皆相等”转换为数组无素之间和的相等关系。这里的关键在于正确地将立方体的8个顶点与数组的8个元素对应。  
可以利用简单的穷举方法建立8个数的全部排列。

**\*程序说明与注释**  
#include<stdio.h>  
#include<stdlib.h>  
int main()  
{  
int a[9],ii=0,i,a1,a2,a3,a4,b1,b2,b3,b4,flag;  
for(i=1;i<=8;i++) /\*输入个数\*/  
{  
printf("Please enter [%d]number:",i);  
scanf("%d",&a[i]);  
ii+=a[i];  
}  
printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/n");  
if(ii%2) /\*和为奇数则输入的8个数不可用\*/  
{  
printf("Sorry they can't be constructed required cube!/n");  
exit(0);  
}  
for(flag=0,a1=1;a1<=8;a1++) /\*flag:完成标记.flag=1;表示完成\*/  
for(a2=1;a2<=8;a2++) /\*采用八重循环建立八个整数的全排列\*/  
if(a2!=a1) /\*前两个数不能相同\*/  
for(a3=1;a3<=8;a3++)   
if(a3!=a2&&a3!=a1) /\*前三个数不能相同\*/  
for(a4=1;a4<=8;a4++)  
if(a4!=a3&&a4!=a2&&a4!=a1) /\*前四个数不能相同\*/  
for(b1=1;b1<=8;b1++)  
if(b1!=a4&&b1!=a3&&b1!=a2&&b1!=a1) /\*不能相同\*/  
for(b2=1;b2<=8;b2++)  
if(b2!=b1&&b2!=a4&&b2!=a3&&b2!=a2&&b2!=a1)  
for(b3=1;b3<=8;b3++)  
if(b3!=b2&&b3!=b1&&b3!=a4&&b3!=a3&&b3!=a2&&b3!=a1)  
/\*不能取相同的数\*/  
for(b4=1;b4<=8;b4++)  
if(b4!=b2&&b4!=b1&&b4!=b3&&b4!=a4&&b4!=a3&&b4!=a2&&b4!=a1)  
if(a[b1]+a[b2]+a[b3]+a[b4]==ii/2  
&&a[a1]+a[a2]+a[b1]+a[b2]==ii/2  
&&a[a1]+a[a4]+a[b1]+a[b4]==ii/2)  
{  
flag=1;goto out; /\*满足条件则将flag置1后退出\*/  
}  
out:  
if(flag)  
{  
printf("They can be constructed required cube as follow:/n");  
printf(" /%2d…………/%2d/n",a[a4],a[a3]);  
printf(" %2d/…………%2d/|/n",a[a1],a[a2]);  
printf(" | | | |/n");  
printf(" | | | |/n");  
printf(" | %2d| | |%2d/n",a[b4],a[b3]);  
printf(" /…………….//n");  
printf(" %2d/………….%2d//n",a[b1],a[b2]);  
}  
else printf("Sorry they can't be constructed required cube!/n");  
}

**\*运行结果**  
Please enter [1] number:20  
Please enter [2] number:45  
Please enter [3] number:39  
Please enter [4] number:25  
Please enter [5] number:29  
Please enter [6] number:7  
Please enter [7] number:3  
Please enter [8] number:2  
Sorry they can't be constructed required cube!

**\*思考题**  
程序中建立全排列的方法效率太低，算法虽然简单但程序过于冗余。请读者自行设计新的算法完成同样的工作。

63.减式还原

编写程序求解下式中各字母所代表的数字，不同的字母代表不同的数字。  
PEAR  
- ARA  
——–  
PEA

**\*问题分析与算法设计**  
类似的问题从计算机算法的角度来说是比较简单的，可以采用最常见的穷举方法解决。程序中采用循环穷举每个字母所可能代表的数字，然后将字母代表的数字转换为相应的整数，代入算式后验证算式是否成立即可解决问题。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int p,e,a,r;  
for(p=1;p<=9;p++) /\*从1到9穷举字母p的全部可能取值\*/  
for(e=0;e<=9;e++) /\*从0到穷举字母e的全部可能取值\*/  
if(p!=e) /\*p不等于e\*/   
for(a=1;a<=9;a++) /\*从0到9穷举字母a的全部可能取值\*/  
if(a!=p&&a!=e)   
for(r=0;r<=9;r++) /\*从0到9穷举字母r的全部可能取值\*/  
if(r!=p&&r!=e&&r!=a&&p\*1000+e\*100+a\*10+r-(a\*100+r\*10+a)  
==p\*100+e\*10+a)  
{  
printf(" PEAR %d%d%d%d/n",p,e,a,r);  
printf(" -ARA - %d%d%d/n",a,r,a);  
printf("……………………./n");  
printf(" PEA %d%d%d/n",p,e,a);  
}  
}

**\*运行结果**  
PEAR 1098  
- ARA - 989  
———- ——  
PEA 109

**\*思考题**  
请复原下面的和式。不同的字母代表不同的数字。  
SEVEN 82524 82526  
THREE 19722 19722  
+ TWO 答案： + 106 + 104  
———- ———– ———–  
TWELVE 102352 102352

 64.乘式还原

A代表数字0到9中的前五个数字，Z代表后五个数字，请还原下列乘式。  
A Z A  
× A A Z  
————  
A A A A  
A A Z Z  
Z A A  
————  
Z A Z A A

**\*问题分析与算法设计**  
问题本身并不复杂，可以对乘式中的每一位使用穷举法，最终可以得到结果。本题的关键在于怎样有效的判断每个部分积的每一位是否满足题意，这一问题处理不好，编写的程序会很长。程序实现中采用了一个判断函数，通过传入函数的标志字符串对所有的数进行统一的判断处理。

**\*程序说明与注释**  
#include<stdio.h>  
void print(long a,long b,long s1,long s2,long s3);  
int jud(long q,char \*pflag);  
int main()  
{  
long i,j,k,l,m,n,term,t1,t2,t3;  
int flag;  
for(i=0;i<=4;++i) /\*被乘数的第一位\*/  
for(j=5;j<=9;++j) /\*被乘数的第二位\*/  
for(k=0;k<=4;++k) /\*被乘数的第三位\*/  
{  
term=100\*i+10\*j+k; /\*被乘数\*/  
for(flag=0,n=0;n<4&&!flag;) /\*乘数的第一位\*/  
flag=jud((t3=++n\*100\*term)/100,"001"); /\*判断第三个部分积\*/  
if(flag)  
{  
for(flag=0,m=0;m<4&&!flag;) /\*乘数的第二位\*/  
flag=jud((t2=++m\*10\*term)/10,"1100"); /\*判断第二个部分积\*/  
if(flag)  
{  
for(flag=0,l=5;l<9&&!flag;) /\*乘数的第三位\*/  
flag=jud(t1=++l\*term,"0000"); /\*判断第一个部分积\*/  
if(flag&&jud(t1+t2+t3,"00101")) /\*判断乘式的积\*/  
print(term,n\*100+m\*10+l,t1,t2,t3);  
}  
}  
}  
}

void print(long a,long b,long s1,long s2,long s3) /\*打印结果\*/   
{  
printf("/n %ld/n",a);  
printf("\*) %ld/n",b);  
printf("…………………./n");  
printf(" %ld/n %ld/n %ld/n",s1,s2/10,s3/100);  
printf("…………………./n");  
printf(" %ld/n",a\*b);  
}  
int jud(long q,char \*pflag) /\*判断一个数的每一位是否满足要求的判断函数\*/  
/\*q:需要判断的数。pflag:标志字符串，A用1表示，Z用0表示。标志串排列顺序：个十百…\*/  
{  
while(q!=0&&\*pflag!=NULL) /\*循环判断对应位的取值范围是否正确\*/  
if(\*pflag-'0'!=(q%10>=5?1:0)) /\*标志位与对应的位不符，返回0\*/  
return 0;  
else   
{  
q/=10;++pflag; /\*若相符则取下一位进行判断\*/  
}  
if(q==0&&\*pflag==NULL) /\*q的位数与标志字符串的长度相同时，返回1\*/  
return 1;  
else return 0;  
}

**\*运行结果**  
3 7 2  
× 2 4 6  
———-  
2 2 3 2  
1 4 8 8  
7 4 4  
————  
9 1 5 1 2

**\*思考题**  
E代表数字0到9中的偶数数字，O代表奇数数字，请还原下列乘式。  
E E O 2 8 5  
× O O 答案 × 3 9  
———– ———–  
E O E O 2 5 6 5  
E O O 8 5 5  
———– ———–  
O O O O O 1 1 1 1 5

65.乘式还原(2)

有乘法算式如下：  
○○○  
× ○○  
————  
○○○○  
○○○○  
————  
○○○○○   
18个○的位置上全部是素数(1、3、5或7)，请还原此算式。

**\*问题分析与算法设计**  
问题中虽然有18数位，但只要确定乘数和被乘数后经过计算就可确定其它的数位。  
乘数和被乘数共有5个数位，要求每个数都是质数。完全可以采用穷举的方法对乘数和被乘数进行穷举，经过判断后找出答案。但是这种方法给人的感觉是“太笨了”，因为组成的数字只是质数(4个)，完全没有必要在那么大的范围内进行穷举，只需要试探每一位数字为质数时的情况即可。  
采用五重循环的方法实现对于5个数字的穷举，前面的许多例题中都已见过。循环实现简单易行，但嵌套的层次太多，需要穷举的变量的数量直接影响到循环嵌套的层数，这种简单的实现方法缺少技巧性。本例的程序中给出了另外一种同样功能的算法，该算法的实现思想请阅读程序。  
程序中并没有直接对质数进行穷举，而是将每个质数与1到4顺序一一对应，在穷举时为处理简单仅对1到4进行穷举处理，待要判断产生的乘积是否满足条件时再利用一个数组完成向对应质数的转换。请体会程序中的处理方法。程序中使用的算法实际上是回朔法。

**\*程序说明与注释**  
#include<stdio.h>  
#define NUM 5 /\*需要穷举的变量数目\*/  
#define C\_NUM 4 /\*每个变量的值的变化范围\*/  
int a[NUM+1]; /\*为需要穷举的变量开辟的数组\*/  
/\*a[1]:被乘数的百位，a[2]：十位,aa[3]:个位 a[4]:被乘数的十位 a[5]:个位\*/  
int b[]={0,2,3,5,7}; /\*存放质数数字的数组，不使用第0号元素\*/  
int f(long sum);

int main()  
{  
int i,not\_finish=1;  
i=2; /\*i:将要进行处理的元素的指针下标。设置初始值\*/  
a[1]=1; /\*为第1号元素设置初始值\*/  
while(not\_finish) /\*not\_finish:程序运行没结束标记\*/  
{  
while(not\_finish&&i<=NUM)  
/\*处理包括第i个元素在内的后续元素，找出当前条件下的一种各个变量的一种可能的取值方法\*/  
if(a[i]>=C\_NUM) /\*当要处理的元素取超过规定的C\_NUM时\*/  
if(i==1&&a[1]==C\_NUM)  
not\_finish=0; /\*若1号元素已经到C\_NUM，则处理全部结束\*/  
else a[i–]=0; /\*将要处理的元素置0，下标-1(回退一个元素)\*/  
else a[i++]++; /\*当前元素值加1后下标指针加1\*/  
if(not\_finish)  
{  
long int sum1,sum2,sum3,sum4; /\*定义临时变量\*/  
sum1=b[a[1>\*100+b[a[2>\*10+b[a[3>; /\*计算被乘数\*/  
/\*利用数组的下标与质数的对应关系完成序号1到4向质数的转换\*/  
sum2=sum1\*b[a[5>; /\*计算乘数个位与被乘数的部分积\*/  
sum3=sum1\*b[a[4>; /\*计算乘数十位与被乘数的部分积\*/  
if(sum2>=2222&&sum2<=7777&&f(sum2)&&sum3>=2222&&sum3<=7777&&f(sum3))  
/\*判断两部分积是否满足题目条件\*/  
if((sum4=sum2+sum3\*10)>=22222&&sum4<=77777&&f(sum4))  
/\*判断乘式的积是否满足题目条件\*/  
{  
printf(" %d/n",sum1); /\*若满足题意，则打印结果\*/   
printf("\* %d%d/n",b[a[4>,b[a[5>);  
printf("……………………/n");  
printf(" %d/n",sum2);  
printf(" %d/n",sum3);  
printf("……………………/n");  
printf(" %d/n",sum4);  
}  
i=NUM; /\*为穷举下一个可能取值作准备\*/  
}  
}  
}  
int f(long sum) /\*判断sum的每一位数字是否是质数，若不是返回0，若是返回1\*/  
{  
int i,k,flag; /\*flag=1:数字是质数的标记\*/  
while(sum>0)   
{  
i=sum%10; /\*取个位的数字\*/  
for(flag=0,k=1;!flag&&k<=C\_NUM;k++)  
if(b[k]==i)  
{  
flag=1;break;  
}  
if(!flag) return 0;  
else sum=sum/10;  
}  
return 1;  
}

**\*运行结果**  
7 7 5  
× 3 3  
———-  
2 3 2 5  
2 3 2 5  
———–  
2 5 5 7 5

**\*思考题**  
以下乘式中，A、B、C代表一确定的数字，○代表任意数字，请复原。  
A B C 2 8 6  
× B A C × 8 2 6  
————- 答案： ————  
○○○○ 1 7 1 6  
○○A 5 7 2  
○○○B 2 2 8 8  
————- —————-  
○○○○○○ 2 3 6 2 3 6

66.除式还原(1)

给定下列除式，其中包含5个7，其它打×的是任意数字，请加以还原。

× 7 × ————–商  
————–  
除数——××| ×××××————-被除数  
×7 7  
————–  
× 7 ×  
× 7 ×  
———-  
× ×  
× ×  
———-  
○

**\*问题分析与算法设计**  
首先分析题目，由除式本身尽可能多地推出已知条件。由除式本身书已知：  
1、被除数的范围是10000到99999，除数的范围是10到99，且可以整除；  
2、商为100到999之间，且十位数字为7；  
3、商的第一位与除数的积为三位数，且后两位为77；  
4、被除数的第三位一定为4；  
5、 7乘以除数的积为一个三位数，且第二位为7；  
6、商的最后一位不能为0，且与除数的积为一个二位数。  
由已知条件就可以采用穷举的方法找出结果。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
long int i;  
int j,l;  
for(i=10000;i<=99999;i++) /\*1. i:被除数\*/  
if(i%1000-i%100==400) /\*4. 被除数的第三位一定为4\*/  
for(j=10;j<=99;j++) /\*1. j: 余数\*/  
if(i%j==0&&(l=i/j)%100>=70&&l%100<80&&l%10!=0&&l>100&&l<=999)  
/\*1. 可以整除&& 2.商l在100到999之间且十位数字为7&&6.商的个数不能为0\*/  
if((j\*(l%10))<100&&j\*(l%10)>10) /\*6. 商的个数与除数的积为二位数\*/  
if(j\*7%100>=70&&j\*7%100<80) /\*5. 7乘以除数的积的第二位为7\*/  
if(j\*(l/100)%100==77&&j\*(l/100)>100)  
/\*商的第一位与除数的积的后两位为77\*/  
printf("%ld/%ld=%d/n",i,j,l);  
}

**\*运行结果**  
51463/53=971。   
可以看作为下列算式：

9 7 1  
————-  
5 3| 5 1 4 6 3  
4 7 7  
————-  
3 7 6  
3 7 1  
———–  
5 3  
5 3  
———–  
○

**\*问题的进一步讨论**  
在推出的已知条件中，几所有的条件都是十分明显的，换句话说，推出的已知条件就是对题目的平铺直叙。这种推已知条件的方法十分简单，并且行之有效。

**\*思考题**  
下列除式中仅给定了一个8，其它打×的位置上是任意数字，请还原。

× 8 × —————-商  
—————-  
除数——-×××| ××××××—————被除数  
××××  
—————  
×××  
×××  
—————  
××××  
××××  
—————  
○

67.除式还原(2)

下列除式中仅在商中给定了一个7，其它打×的位置全部是任意数字，请还原。

×7××× ————-商  
——————   
除数 ——————-×××| ×××××××× ————-被除数  
×××× ————-1)  
—————   
××× ————-2)  
××× ————-3)  
—————  
×××× ————-4)  
××× ————-5)  
—————–  
×××× ————-6)  
×××× ————-7)  
—————–  
0

**\*问题分析与算法设计**  
这道题是不可能用单纯的穷举法求解的，一则计算时间太长，二则难于求出除式中各部分的值。  
对除式进行分析，改可能多地推出限制条件：  
由3)可以看出，商的第二位7乘除数得一个三位数，所以除数<=142。  
由除数乘商的第一位为一个四位数可知，商的第一位只能为8或9且除数>=112。同时商的第五位也为8或9数的前四位一定<=142\*9+99且>=1000+10。  
由4)、5)、6)可以看出，4)的前两位一定为“10”；5)的第一位一定为“9”；6)的前两位一定在10到99之间；商的第四位一定为为0。  
由 5)的第一位一定是“9”和“112”<=除数<=142可知：商的第三位可能为7或8。  
由除式本身可知：商的第四位为0。  
由 1)可知：除数X商的第一位应当为一个四位数。  
由 5)可知：除数X商的第三位应当为一个三位数。  
编程时为了方便，将被除数分解：前四位用a[0]表示，第五位用a[1]，第六位用a[2]，第七八两位用a[3];除数用变量b表示；分解商：第一位用c[0]，第五位用c[2];其它的部分商分别表示为：2)的前两位为d[0]，4)的前三位为d[1]，6)的前二位为d[2]。将上述分析用数学的方法综合起来可以表示为：  
被除数： 1010<=a[0]<=1377 0<=a[1]<=9  
0<=a[2]<=9 0<=a[3]<=99  
除数： 112<=b <=142  
商： 8<=c[0]<=9 7<=c[1]<=8 8<=c[2]<=9  
2)的前两位： 10<=d[0]<=99  
4)的前三位： 100<=d[1]<b  
6)的前两位： 10<=d[2]<=99  
1)式部分积： b\*c[0]>1000  
5)式部分积： 100<b\*c[1]<1000

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int a[4],b,c[3],d[4],i=1;  
for(a[0]=1010;a[0]<=1377;a[0]++)  
for(b=112;b<=142;b++)  
for(c[0]=8;c[0]<=9;c[0]++)  
if(b\*c[0]>1000&&(d[0]=a[0]-b\*c[0])>=10&&d[0]<100)  
for(a[1]=0;a[1]<=9;a[1]++)  
if((d[1]=d[0]\*10+a[1]-b\*7)>=100&&d[1]<b)  
for(a[2]=0;a[2]<=9;a[2]++)  
for(c[1]=7;c[1]<=8;c[1]++)  
if(b\*c[1]<1000&&(d[2]=d[1]\*10+a[2]-b\*c[1])>=10&&d[2]<100)  
for(a[3]=0;a[3]<=99;a[3]++)  
for(c[2]=8;c[2]<=9;c[2]++)  
if(d[2]\*100+a[3]-b\*c[2]==0)  
{  
printf("No%2d:",i++);  
printf("%d%d%d%d%d/",a[0],a[1],a[2],a[3]/10,a[3]%10);  
printf("%d=",b);  
printf("%d%d%d%d%d/n",c[0],7,c[1],0,c[2]);  
}  
}  
\*运行结果：  
No 1:12128316/124=97809

**\*思考题**  
下列除式中“×”所在的位置全部是任意数字，请还原。  
×××××  
——————-  
××× | ××××××××  
××××  
——————  
××××  
×××  
—————  
×××  
×××  
———–  
××××  
××××  
———–  
0

68.九位累进可除数

求九位累进可除数。所谓九位累进可除数就是这样一个数：这个数用到1到9这九个数字组成，每个数字刚好只出现一次。这九个位数的前两位能被2整除，前三位能被3整除……前N位能被N整除，整个九位数能被9整除。

**\*问题分析与算法设计**  
问题本身可以简化为一个穷举问题：只要穷举每位数字的各种可能取值，按照题目的要求对穷举的结果进行判断就一定可以得到正确的结果。  
问题中给出了“累进可除”这一条件，就使得我们可以在穷举法中加入条件判断。在穷举的过程中，当确定部分位的值后，马上就判断产生的该部分是否符合“累进可除”条件，若符合，则继续穷举下一位数字；否则刚刚产生的那一位数字就是错误的。这样将条件判断引入到穷举法之中，可以尽可能早的发现矛盾，尽早地放弃不必要穷举的值，从而提高程序的执行效率。  
为了达到早期发现矛盾的目的，不能采用多重循环的方法实行穷举，那样编出的程序质量较差。程序中使用的算法不再是穷举法，而是回朔法。

**\*程序说明与注释**  
#include<stdio.h>  
#define NUM 9  
int a[NUM+1];  
int main()  
{  
int i,k,flag,not\_finish=1;  
long sum;  
i=1;  
/\*i:正在处理的数组元素，表示前i-1个元素已经满足要求，正处理的是第i个元素\*/  
a[1]=1; /\*为元素a[1]设置初值\*/  
while(not\_finish) /\*not\_finish=1:处理没有结束\*/  
{  
while(not\_finish&&i<=NUM)  
{  
for(flag=1,k=1;flag&&k<i;k++)  
if(a[k]==a[i])flag=0; /\*判断第i个元素是否与前i-1个元素重复\*/  
for(sum=0,k=1;flag&&k<=i;k++)  
{  
sum=10\*sum+a[k];  
if(sum%k)flag=0; /\*判断前k位组成的整数是否能被k整除\*/  
}  
if(!flag) /\*flag=0:表示第i位不满足要求，需要重新设置\*/  
{  
if(a[i]==a[i-1]) /\*若a[i]的值已经经过一圈追上a[i-1]\*/  
{  
i–; /\*i值减1，退回处理前一个元素\*/  
if(i>1&&a[i]==NUM)  
a[i]=1; /\*当第i位的值达到NUM时，第i位的值取1\*/  
else if(i==1&&a[i]==NUM) /\*当第1位的值达到NUM时结束\*/  
not\_finish=0; /\*置程序结束标记\*/  
else a[i]++; /\*第i位的值取下一个,加1\*/  
}  
else if(a[i]==NUM) a[i]=1;  
else a[i]++;  
}  
else /\*第i位已经满足要求，处理第i+1位\*/  
if(++i<=NUM) /\*i+1处理下一元素，当i没有处理完毕时\*/  
if(a[i-1]==NUM) a[i]=1; /\*若i-1的值已为NUM，则a[i]的值为1\*/  
else a[i]=a[i-1]+1; /\*否则，a[i]的初值为a[i-1]值的"下一个"值\*/  
}  
if(not\_finish)  
{  
printf("/nThe progressire divisiable number is:");  
for(k=1;k<=NUM;k++) /\*输出计算结果\*/  
printf("%d",a[k]);  
if(a[NUM-1]<NUM) a[NUM-1]++;  
else a[NUM-1]=1;  
not\_finish=0;  
printf("/n");  
}  
}  
}

**\*运行结果**  
The progressire divisible number is: 381654729

**\*思考题**  
求N位累进可除数。用1到9这九个数字组成一个N(3<=N<=9)位数，位数字的组成不限，使得该N位数的前两位能被2整除，前3位能被3整除，……，前N位能被N整除。求满足条件的N位数。

69.魔术师的猜牌术(1)

魔术师利用一副牌中的13张黑桃，预先将它们排好后迭在一起，牌面朝下。对观众说：我不看牌，只数数就可以猜到每张牌是什么，我大声数数，你们听，不信？你们就看。魔术师将最上面的那张牌数为1，把它翻过来正好是黑桃A，将黑桃A放在桌子上，然后按顺序从上到下数手上的余牌，第二次数1、2，将第一张牌放在这迭牌的下面，将第二张牌翻过来，正好是黑桃2，也将它放在桌子上，第三次数1、2、3，将前面两张依次放在这迭牌的下面，再翻第三张牌正好是黑桃3。这样依次进行将13张牌全翻出来，准确无误。问魔术师手中的牌原始顺序是怎样安排的？

**\*问题分析与算法设计**  
题目已经将魔术师出牌的过程描述清楚，我们可以利用倒推的方法，很容易地推出原来牌的顺序。  
人工倒推的方法是：在桌子上放13空盒子排成一圈，从1开始顺序编号，将黑桃A放入1号盒子中，从下一个空盒子开始对空的盒子计数，当数到第二个空盒子时，将黑桃2放入空盒子中，然后再从下一个空盒子开始对空盒子计数，顺序放入3、4、5…，直到放入全部3张牌。注意在计数时要跳过非空的盒子，只对空盒子计数。最后牌在盒子中的顺序，就是魔术师手中原来牌的顺序。  
这种人工的方法是行之有效的，计算机可以模拟求解。

**\*程序说明与注释**  
#include<stdio.h>  
int a[14];  
int main()  
{  
int i,n,j=1; /\*j:数组(盒子)下标，初始时为1号元素\*/  
printf("The original order of cards is:");  
for(i=1;i<=13;i++) /\*i:要放入盒子中的牌的序号\*/  
{  
n=1;  
do{  
if(j>13) j=1; /\*由于盒子构成一个圈，j超过最后一个元素则指向1号元素\*/  
if(a[j]) j++; /\*跳过非空的盒子，不进行计数\*/  
else{ if(n==i) a[j]=i; /\*若数到第i个空盒子，则将牌放入空盒中\*/  
j++;n++; /\*对空盒计数，数组下标指向下一个盒子\*/  
}   
}while(n<=i); /\*控制空盒计数为i\*/  
}  
for(i=1;i<=13;i++) /\*输出牌的排列顺序\*/  
printf("%d ",a[i]);  
printf("/n");  
}

**\*运行结果**  
The original order of cards is:1 8 2 5 10 3 12 11 9 4 7 6 13

70.魔术师的猜牌术(2)

魔术师再次表演，他将红桃和黑桃全部迭在一起，牌面朝下放在手中，对观众说：最上面一张是黑桃A，翻开后放在桌上。以后，从上至下每数两张全依次放在最底下，第三张给观众看，便是黑桃2，放在桌上后再数两张依次放在最底下，第三张给观众看，是黑桃3。如此下去，观众看到放在桌子上牌的顺序是：  
黑桃 A 2 3 4 5 6 7 8 9 10 J Q K  
红桃 A 2 3 4 5 6 7 8 9 10 J Q K  
问魔术师手中牌的原始顺序是什么？

**\*问题分析与算法设计**  
本题可在上题的基础上进行编程，不同的在于计数的方法和牌的张数，这些并不影响我们求解题目的思路，仍可按照倒推的方法，得到原来魔术师手中的牌的顺序。

**\*程序说明与注释**  
#include<stdio.h>  
int a[27];  
int main()  
{  
int i,n,j=1;  
a[1]=1; /\*初始化第一张牌\*/  
printf("The original order of cards is:(r:rad b:block):/n");  
for(i=2;i<=26;i++)  
{  
n=1;  
do{   
if(j>26) j=1; /\*超过最后一个元素则指向1号元素\*/  
if(a[j]) j++; /\*跳过非空的盒子，不进行计数\*/  
else{  
if(n==3) a[j]=i; /\*若数到第3个空盒子，则将牌放入空盒中\*/  
j++; n++; /\*对空盒计数，数组下标指向下一个盒子\*/  
}  
}while(n<=3); /\*控制空盒计数为3\*/  
}  
for(i=1;i<=26;i++) /\*输出牌的排列顺序\*/  
{  
printf("%c",a[i]>13? 'r':'b');  
printf("%d ",a[i]>13? a[i]-13:a[i]);  
if(i==13) printf("/n");  
}  
printf("/n");  
}

**\*运行结果**  
The original order of cards is:(r:rad b:black):  
b1 r6 b10 b2 r12 r3 b3 b11 r9 b4 r7 b12 b5  
r4 r13 b6 b13 r11 b7 r5 r1 b8 r8 r10 b9 r2

 71.约瑟夫问题

这是17世纪的法国数学家加斯帕在《数目的游戏问题》中讲的一个故事：15个教徒和15 个非教徒在深海上遇险，必须将一半的人投入海中，其余的人才能幸免于难，于是想了一个办法：30个人围成一圆圈，从第一个人开始依次报数，每数到第九个人就将他扔入大海，如此循环进行直到仅余15个人为止。问怎样排法，才能使每次投入大海的都是非教徒。

**\*问题分析与算法设计**  
约瑟夫问题并不难，但求解的方法很多；题目的变化形式也很多。这里给出一种实现方法。  
题目中30个人围成一圈，因而启发我们用一个循环的链来表示。可以使用结构数组来构成一个循环链。结构中有两个成员，其一为指向下一个人的指针，以构成环形的链；其二为该 人是否被扔下海的标记，为1表示还在船上。从第一个人开始对还未扔下海的人进行计数，每数到9时，将结构中的标记改为0，表示该人已被扔下海了。这样循环计数直到有15个人被扔下海为止。

**\*程序说明与注释**  
#include<stdio.h>  
struct node  
{  
int nextp; /\*指向下一个人的指针(下一个人的数组下标)\*/  
int no\_out; /\*是否被扔下海的标记。1：没有被扔下海。0：已被扔下海\*/  
}link[31]; /\*30个人，0号元素没有使用\*/  
int main()  
{  
int i,j,k;  
printf("The original circle is(+:pagendom,@:christian):/n");  
for(i=1;i<=30;i++) /\*初始化结构数组\*/  
{  
link[i].nextp=i+1; /\*指针指向下一个人(数组元素下标)\*/  
link[i].no\_out=1; /\*标志置为1，表示人都在船上\*/  
}  
link[30].nextp=1; /\*第30个人的指针指向第一个人以构成环\*/  
j=30; /\*j:指向已经处理完毕的数组元素，从link[i]指向的人开始计数\*/  
for(i=0;i<15;i++) /\*i:已扔下海的人数计数器\*/  
{  
for(k=0;;) /\*k:决定哪个人被扔下海的计数器\*/  
if(k<15)  
{  
j=link[j].nextp; /\*修改指针，取下一个人\*/  
k+=link[j].no\_out; /\*进行计数。因已扔下海的人计标记为0\*/  
}  
else break; /\*计数到15则停止计数\*/  
link[j].no\_out=0; /\*将标记置 0，表示该人已被扔下海\*/  
}  
for(i=1;i<=30;i++) /\*输出结果\*/  
printf("%c",link[i].no\_out? '@':'+'); /\*+:被扔下海， @：在船上\*/  
printf("/n");  
}

**\*运行结果**  
The original circle is(+:pagandom, @:christian):  
+++@@+@+@@@+@+++@@+@@@+++@+@@+  
(+"表示被扔下海海的非教徒 @：留在船上活命的教徒)

**\*思考题**  
有N个小孩围 成一圈并依次编号，教师指定从第M个小孩开始报数，报到第S个小孩即令其出列。然后从下一个孩子继续报数，数到第S个小孩又令其出列，如此直到所有的孩子都出列。求小孩出列的先后顺序。

72.邮票组合

某人有四张3分的邮票和三张5分的邮票，用这些邮票中的一张或若干张可以得到多少种不同的邮资？  
\*问题分析与算法设计   
将问题进行数学分析，不同张数和面值的邮票组成的邮资可用下列公式计算：  
S=3\*i+5\*j  
其中i为3分邮柰的张数，j为5分的张数  
按题目的要求，3分的邮票可以取0、1、2、3、4张，5分的邮票可以取0、1、2、3张。采用穷举方法进行组合，可以求出这些不同面值不同张数的邮标组合后的邮资。

**\*程序说明与注释**  
#include<stdio.h>  
int a[27];  
int main()  
{  
int i,j,k,s,n=0;  
for(i=0;i<=4;i++) /\*i:取三分邮票的张数\*/  
for(j=0;j<=3;j++) /\*j:取5分邮票的张数\*/  
{  
s=i\*3+j\*5; /\*计算组成的邮票面值\*/  
for(k=0;a[k];k++) /\*查找是否有相同的邮资\*/  
if(s==a[k])break;  
if(!a[k]&&s) /\*没有找到相同的邮资则满足要求存入数组\*/  
{  
a[k]=s; n++;  
}  
}  
printf("%d kinds:",n); /\*输出结果\*/  
for(k=0;a[k];k++)  
printf("%d ",a[k]);  
printf("/n");  
}

**\*运行结果**  
19 kinds: 5 10 15 3 8 13 18 6 11 16 21 9 14 19 24 12 17 22 27

 73.和数能表示1~23的5个正整数

已知五个互不相同的正整数之和为23，且从这五个数中挑选若干个加起来可以表示从1到23之内的全部自然数。问这五个数是什么？

**\*问题分析与算法设计**  
从计算机程序设计的角度来说，可以用穷举法分解23，然后判断所分解的五个数是否可以表示1到23 之间的全部整数。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int a,b,c,d,e,i,j,k,l,m,x,count=0,f=0; /\*f:分解的5个数可以表示出1~23的标记\*/  
printf("There are following possble result:/n");  
for(a=1;a<=23;a++) /\*将23分解为a,b,c,d,e五个数\*/  
for(b=1+a;b<=23-a;b++)  
for(c=1+b;c<=23-a-b;c++)  
for(d=1+c;d<=23-a-b-c;d++)  
{  
f=1;  
if((e=23-a-b-c-d)>d)  
for(f=0,x=1;x<24&&!f;x++) /\*判断5个数可否表示1~23\*/  
for(f=1,i=0;i<2&&f;i++) /\*穷举5个数的全部取舍\*/   
for(j=0;j<2&&f;j++)  
for(k=0;k<2&&f;k++)  
for(l=0;l<2&&f;l++)  
for(m=0;m<2&&f;m++)  
if(x==a\*i+b\*j+c\*k+d\*l+e\*m) f=0;  
if(!f) printf("[%d]: %d %d %d %d %d/n",++count,a,b,c,d,e);  
}  
}

**\*运行结果**  
There are following possble result:  
[1]: 1 2 3 5 12  
[2]: 1 2 3 6 11  
[3]: 1 2 3 7 10  
[4]: 1 2 4 5 11  
[5]: 1 2 4 6 10  
[6]: 1 2 4 7 9

74.可称1~40磅的4块砝码

法国数学家梅齐亚克在他著名的《数字组合游戏》(1962)中提出了一个问题：一位商人有一个重40磅的砝码，一天不小心将砝码摔成了四块。后来商人称得每块的重量都是整磅数，而且发现这四块碎片可以在天平上称1至40磅之间的任意重量。请问这四块碎片各重多少？

**\*问题分析与算法设计**  
本题是上一题的发展。题目中给出的条件是“在天平上”，这意味着：同一砝码既可以放在天平的左侧，也可以放在天平的右侧。若规定重物只能放在天平的左侧，则当天平平衡时有：  
重物重量+左侧砝码重量总和=右侧砝码重量总和  
由此可得：  
重物重量=右侧砝码重量总和-左侧砝码重量总和  
编程时只要根据以上公式，使“右侧砝码重量总和-左侧砝码重量总和”可以表示1到40之间的全部重量即可。编程中要注意的是：怎样采用一种简单的方法来表示一个砝码是在天平的左侧还是在天平的右侧，或是根本没有使用。  
以下程序采用1、 -1和0分别表示上述三种情况，请注意理解。

**\*程序说明与注释**  
#include<stdio.h>  
#include<math.h>  
int main()  
{  
int weight1,weight2,weight3,weight4,d1,d2,d3,d4,x,flag; /\*flag:满足题意的标记\*/  
printf("The weight is broke up as following 4 pieces:");  
for(weight1=1;weight1<=40;weight1++) /\*将40分解成4份\*/  
for(weight2=weight1+1;weight2<=40-weight1;weight2++)  
for(weight3=weight2+1;weight3<=40-weight1-weight2;weight3++)  
if((weight4=40-weight1-weight2-weight3)>=weight3)  
{  
for(flag=1,x=1;x<41&&flag;x++) /\*判断可否称出1~40之间的全部重量\*/  
for(flag=0,d1=1;d1>-2;d1–) /\*将重物放在天平的左边\*/  
for(d2=1;d2>-2&&!flag;d2–) /\*1:砝码在天平右边\*/  
for(d3=1;d3>-2&&!flag;d3–) /\*0：不用该砝码\*/  
for(d4=1;d4>-2&&!flag;d4–) /\*-1:砝码在天平的左边\*/  
if(x==weight1\*d1+weight2\*d2+weight3\*d3+weight4\*d4)  
flag=1;  
if(flag) printf("%d %d %d %d/n",weight1,weight2,weight3,weight4);  
flag=0;   
}  
}

**\*运行结果**  
The weight is broke up as following 4 pieces: 1 3 9 27

75.10个小孩分糖果

十个小孩围成一圈分糖果，老师分给第一个小孩10块，第二个小孩2块，第三个小孩8块，第四个小孩22块，第五个小孩16块，第六个小孩4块，第七个小孩10块，第八个小孩6块，第九个小孩14块，第十个小孩20块。然后所有的小孩同时将手中的糖分一半给右边的小孩；糖块数为奇数的人可向老师要一块。问经过这样几次后大家手中的糖的块数一样多？每人各有多少块糖？

**\*问题分析与算法设计**  
题目描述的分糖过程是一个机械的重复过程，编程算法完全可以按照描述的过程进行模拟。

**\*程序说明与注释**  
#include<stdio.h>  
void print(int s[]);  
int judge(int c[]);  
int j=0;  
int main()  
{  
static int sweet[10]={10,2,8,22,16,4,10,6,14,20}; /\*初始化数组数据\*/  
int i,t[10],l;  
printf(" child/n");  
printf(" round 1 2 3 4 5 6 7 8 9 10/n");  
printf("………………………../n");  
print(sweet); /\*输出每个人手中糖的块数\*/  
while(judge(sweet)) /\*若不满足要求则继续进行循环\*/  
{   
for(i=0;i<10;i++) /\*将每个人手中的糖分成一半\*/  
if(sweet[i]%2==0) /\*若为偶数则直接分出一半\*/  
t[i]=sweet[i]=sweet[i]/2;  
else /\*若为奇数则加1后再分出一半\*/  
t[i]=sweet[i]=(sweet[i]+1)/2;  
for(l=0;l<9;l++) /\*将分出的一半糖给右(后)边的孩子\*/  
sweet[l+1]=sweet[l+1]+t[l];  
sweet[0]+=t[9];  
print(sweet); /\*输出当前每个孩子中手中的糖数\*/  
}  
}  
int judge(int c[])  
{  
int i;  
for(i=0;i<10;i++) /\*判断每个孩子手中的糖是否相同\*/  
if(c[0]!=c[i]) return 1; /\*不相同返回 1\*/  
return 0;  
}  
void print(int s[]) /\*输出数组中每个元素的值\*/  
{  
int k;  
printf(" %2d ",j++);  
for(k=0;k<10;k++) printf("%4d",s[k]);  
printf("/n");  
}

76.小明买书

小明假期同爸爸一起去书店，他选中了六本书，每本书的单价分别为：3.1，1.7，2，5.3，0.9和7.2。不巧的是，小明的爸爸只带了十几块钱，为了让小明过一个愉快的假期，爸爸扔然同意买书，但提邮购一个要求，要小明从六本书中选出若干本，使得单价相加所得的和同10最接近。你能够帮助小明解决这个问题吗？

**\*问题分析与算法设计**  
分析题意，可将题目简化为：从六个数中选出若干个求和，使得和与10的差值最小。  
题目中隐含两个问题，其一是怎样从六个数中选出若干个数；其二是求与10的差。  
从六个数中选出若干个数实质是从六个数中选出若干个进行组合。每个数在组合过程中只有两种情况：要么是选中参加求和，要么是没选中不参加求和。这样就可以使用六重循环对每个数是否参加求和进行全部可能情况的组合。  
关于求与10的差值应当注意的是：差值的含义是指差的绝对值。例如：“9-10=-1"和"11-10=1",但9和11这两者与10的差值都是1。若认为”9“与”10的差值为-1就错了。

**\*程序说明与注释**  
#include<stdio.h>  
#include<math.h>  
int main()  
{  
int d[6],m,i,j;  
long b[63],flag;  
float c[6],min,x;  
printf("Please enter the prices of 6 books:");  
for(i=0;i<6;i++) scanf("%f",&c[i]); /\*输入六个浮点数\*/  
for(i=0,min=-1,d[0]=0;d[0]<2;d[0]++) /\*建立六个数的全部组合并处理\*/  
for(d[1]=0;d[1]<2;d[1]++) /\*i:差值具有min组合的数量\*/  
for(d[2]=0;d[2]<2;d[2]++) /\*min:与10的最小差值\*/  
for(d[3]=0;d[3]<2;d[3]++) /\*d[]:组合时是否取该数的标志\*/  
for(d[4]=0;d[4]<2;d[4]++)  
for(d[5]=0;d[5]<2;d[5]++)  
{  
for(flag=0,x=0,j=5;j>=0;j–)   
/\*flag:将六个数的组合用对应的一个十进制位表示 x:对应六个数组合的和\*/  
{  
x+=c[j]\*d[j]; flag=flag\*10+d[j];  
}  
x=((x-10>0)? x-10:10-x); /\*x: 组合的和与10的差\*/  
if(min<0)  
{  
min=x; /\*对第一次计算出的差min进行处理\*/  
b[i++]=flag; /\*b[]:有相同的min的flag的数组 i:b[]数组的下标\*/  
}  
else if(min-x>1.e-6) /\*对新的min的处理\*/  
{  
min=x; b[0]=flag; i=1;  
}  
else if(fabs((double)x-min)<1.e-6)  
b[i++]=flag; /\*对相等min的处理\*/  
}  
for(m=0;m<i;m++) /\*输出全部i个与10的差值均为min的组合\*/  
{  
printf("10(+ -)%.2f=",min);  
for(flag=b[m],j=0;flag>0;j++,flag/=10)  
if(flag%10) /\*将b[]中存的标记flag还原为各个数的组合\*/  
if(flag>1) printf("%.2f+",c[j]);  
else printf("%.2f/n",c[j]);  
}  
}

**\*运行结果**  
Please enter the prices of 6 books:3.1 1.7 2.0 5.3 0.9 7.2  
10(+ -)0.10=2.00+0.90+7.20  
10(+ -)0.10=1.70+2.00+5.30+0.90  
10(+ -)0.10=3.10+1.70+5.30

**\*思考题**  
可以看出，程序中求六个数所能产生全部组合的算法并不好，使用六重循环进行处理使程序显得不够简洁。可以设计出更通用、优化的算法产生全部组合。

77.波松瓦酒的分酒趣题

法国著名数学家波瓦松在表年时代研究过一个有趣的数学问题：某人有12品脱的啤酒一瓶，想从中倒出6品脱，但他没有6品脱的容器，仅有一个8品脱和5品脱的容器，怎样倒才能将啤酒分为两个6品脱呢？

**\*问题分析与算法设计**  
将12品脱酒 8品脱和5品脱的空瓶平分，可以抽象为解不定方程：  
8x-5y=6  
其意义是：从12品脱的瓶中向8品脱的瓶中倒x次，并且将5品脱瓶中的酒向12品脱的瓶中倒y次，最后在12品脱的瓶中剩余6品脱的酒。  
用a,b,c代表12品脱、8品脱和5品脱的瓶子，求出不定方程的整数解，按照不定方程的意义则倒法为：  
a -> b -> c ->a  
x y  
倒酒的规则如下：  
1) 按a -> b -> c ->a的顺序；  
2) b倒空后才能从a中取  
3) c装满后才能向a中倒  
按以上规则可以编写出程序如下：

**\*程序说明与注释**  
#include<stdio.h>  
void getti(int a,int y,int z);  
int i; /\*最后需要分出的重量\*/  
int main()  
{  
int a,y,z;  
printf("input Full a,Empty b,c,Get i:"); /\*a 满瓶的容量 y:第一个空瓶的容量 z:第二个空瓶的容量\*/  
scanf("%d%d%d%d",&a,&y,&z,&i);  
getti(a,y,z); /\*按a -> y -> z -> a的操作步骤\*/  
getti(a,z,y); /\*按a -> z -> y -> a的步骤\*/  
}  
void getti(int a,int y,int z) /\*a:满瓶的容量 y:第一个空瓶的容量 z:第二个空瓶的容量\*/  
{  
int b=0,c=0; /\* b:第一瓶实际的重量 c:第二瓶实际的重量\*/  
printf(" a%d b%d c%d/n %4d%4d%4d/n",a,y,z,a,b,c);  
while(a!=i||b!=i&&c!=i) /\*当满瓶!=i或另两瓶都!=i\*/  
{  
if(!b)  
{ a-=y; b=y;} /\*如果第一瓶为空，则将满瓶倒入第一瓶中\*/  
else if(c==z)  
{ a+=z; c=0;} /\*如果第二瓶满，则将第二瓶倒入满瓶中\*/  
else if(b>z-c) /\*如果第一瓶的重量>第二瓶的剩余空间\*/  
{ b-=(z-c);c=z;} /\*则将装满第二瓶，第一瓶中保留剩余部分\*/  
else{ c+=b; b=0;} /\*否则，将第一瓶全部倒入第二瓶中\*/  
printf(" %4d %4d %4d/n",a,b,c);  
}  
}

**\*思考题**  
上面的程序中仅给出了两种分酒的方法，并没有找出全部的方法。请设计新的算法，找出全部的分酒方法，并找出一种倒酒次数最少的方法。

78.求π的近似值

请利用“正多边形逼近”的方法求出π的近似值

**\*问题分析与算法设计**  
利用“正多边形逼近”的方法求出π值在很早以前就存在，我们的先人祖冲之就是用这种方法在世界上第一个得到精确度达小数点后第6位的π值的。  
利用圆内接正六边形边长等于半径的特点将边数翻番，作出正十二边形，求出边长，重复这一过程，就可获得所需精度的π的近似值。  
假设单位圆内接多边形的边长为2b，边数为i，则边数加倍后新的正多边形的边长为：  
x=√──────  
2-2\*√───  
1-b\*b  
──────  
2  
周长为：   
y=2 \* i \* x i:为加倍前的正多边形的边数

**\*程序说明与注释**  
#include<stdio.h>  
#include<math.h>  
int main()  
{  
double e=0.1,b=0.5,c,d;  
long int i; /\*i: 正多边形边数\*/  
for(i=6;;i\*=2) /\*正多边形边数加倍\*/  
{  
d=1.0-sqrt(1.0-b\*b); /\*计算圆内接正多边形的边长\*/  
b=0.5\*sqrt(b\*b+d\*d);  
if(2\*i\*b-i\*e<1e-15) break; /\*精度达1e-15则停止计算\*/  
e=b; /\*保存本次正多边形的边长作为下一次精度控制的依据\*/  
}  
printf("pai=%.15lf/n",2\*i\*b); /\*输出π值和正多边形的边数\*/  
printf("The number of edges of required polygon:%ld/n",i);  
}

**\*运行结果**  
pai=3.141592653589794  
The number of edges of required polygon:100663296

**\*思考题**  
请用外切正多边形逼近的方法求π的近似值。

79.求π的近似值(2)

利用随机数法求π的近似值

**\*问题分析与算法设计**  
随机数法求π的近似值的思路：在一个单位边长的正方形中，以边长为半径，以一个顶点为圆心，在政权方形上作四分之一圆。随机的向正方形内扔点，若落入四分之一圆内则计数。重复向正方形内扔足够多的点，将落入四分之一圆内的计数除以总的点数，其值就是π值四分之一的近似值。  
按此方法可直接进行编程，注意：本方法求出的π值只有统计次数足够多时才可能准确。

**\*程序说明与注释**  
#include<time.h>  
#include<stdlib.h>  
#include<stdio.h>  
#define N 30000  
int main()  
{  
float x,y;  
int c=0,d=0;  
randomize();  
while(c++<=N)  
{  
x=random(101); /\*x:坐标。产生0到100之间共101个的随机数\*/  
y=random(101); /\*y:坐标。产生0到100之间共101个的随机数\*/  
if(x\*x+y\*y<=10000) /\*利用圆方程判断点是否落在圆内\*/  
d++;  
}  
printf(" pi=%f/n",4. \*d/N); /\*输出求出的π值\*/  
}

**\*运行结果**  
多次运行程序，可能得到多个不同的对口果，这是因为采用的是统计规律求出的近似值，只有当统计的次数足够大时，才可能逼近π值。运行四次，可能的结果是：  
3.122267  
3.139733  
3.133733

80.奇数平方的一个有趣性质

编程验证“大于1000的奇数其平方与1的差是8的倍数”。

**\*问题分析与算法设计**  
本题是一个很容易证明的数学定理，我们可以编写程序验证它。  
题目中给出的处理过程很清楚，算法不需要特殊设计。可以按照题目的叙述直接进行验证(程序中仅验证到3000)。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
long int a;  
for(a=1001;a<=3000;a+=2)  
{  
printf("%ld:",a); /\*输出奇数本身\*/  
printf("(%ld\*%ld-1)/8",a,a); /\*输出(奇数的平方减1)/8\*/  
printf("=%ld",(a\*a-1)/8); /\*输出被8除后的商\*/  
printf("+%ld/n",(a\*a-1)%8); /\*输出被8除后的余数\*/  
}

}

81.角谷猜想

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，于是产生角谷猜想。猜想的内容是：任给一个自然数，若为偶数除以2，若为奇数则乘3加1，得到一个新的自然数后按照上面的法则继续演算，若干次后得到的结果必然为1。请编程验证。

**\*问题分析与算法设计**  
本题是一个沿未获得一般证明的猜想，但屡试不爽，可以用程序验证。  
题目中给出的处理过程很清楚，算法不需特殊设计，可按照题目的叙述直接进行证。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int n,count=0;  
printf("Please enter number:");  
scanf("%d",&n); /\*输入任一整数\*/  
do{  
if(n%2)  
{  
n=n\*3+1; /\*若为奇数，n乘3加1\*/  
printf("[%d]:%d\*3+1=%d/n",++count,(n-1)/3,n);  
}  
else  
{  
n/=2; /\*若为偶数n除以2\*/  
printf("[%d]: %d/2=%d/n",++count,2\*n,n);  
}  
}while(n!=1); /\*n不等于1则继续以上过程\*/  
}

82.四方定理

数论中著名的“四方定理”讲的是：所有自然数至多只要用四个数的平方和就可以表示。  
请编程证此定理。

**\*问题分析与算法设计**  
本题是一个定理，我们不去证明它而是编程序验证。  
对四个变量采用试探的方法进行计算，满足要求时输出计算结果。

**\*程序说明与注释**  
#include<stdio.h>  
#include<stdlib.h>  
int main()  
{  
int number,i,j,k,l;  
printf("Please enter a number=");  
scanf("%d",&number); /\*输入整数\*/  
for(i=1;i<number/2;i++) /\*试探法。试探i,j,k,k的不同值\*/  
for(j=0;j<=i;j++)  
for(k=0;k<=j;k++)  
for(l=0;l<=k;l++)  
if(number==i\*i+j\*j+k\*k+l\*l) /\*若满足定理要求则输出结果\*/  
{  
printf(" %d=%d\*%d+%d\*%d+%d\*%d+%d\*%d/n",number,i,i,j,j,k,k,l,l);  
exit(0);  
}  
}

**\*运行结果**  
1) Please enter a number = 110  
110=7\*7+6\*6+4\*4+3\*3  
2) Please enter a number = 211  
211=8\*8+7\*7+7\*7+7\*7  
3) Please enter a number = 99  
99=7\*7+5\*5+4\*4+3\*3

83.卡布列克常数

验证卡布列克运算。任意一个四位数，只要它们各个位上的数字是不全相同的，就有这样的规律：  
1)将组成该四位数的四个数字由大到小排列，形成由这四个数字构成的最大的四位数；  
2)将组成该四位数的四个数字由小到大排列，形成由这四个数字构成的最小的四位数(如果四个数中含有0，则得到的数不足四位)；  
3)求两个数的差，得到一个新的四位数(高位零保留)。  
重复以上过程，最后得到的结果是6174，这个数被称为卡布列克数。

**\*问题分析与算法设计**  
题目中给出的处理过程很清楚，算法不需要特殊设计，可按照题目的叙述直接进行验证。

**\*程序说明与注释**  
#include<stdio.h>  
void vr6174(int);  
void parse\_sort(int num,int \*each);  
void max\_min(int \*each,int \*max,int \*min);  
void parse\_sort(int num,int \*each);  
int count=0;  
int main()  
{  
int n;  
printf("Enter a number:");  
scanf("%d", &n); /\*输入任意正整数\*/  
vr6174(n); /\*调用函数进行验证\*/  
}

void vr6174(int num)  
{  
int each[4],max,min;  
if(num!=6174&&num) /\*若不等于74且不等于0则进行卡布列克运算\*/  
{  
parse\_sort(num,each); /\*将整数分解，数字存入each数组中\*/  
max\_min(each,&max,&min); /\*求数字组成的最大值和最小值\*/   
num=max-min; /\*求最大值和最小值的差\*/  
printf("[%d]: %d-%d=%d/n",++count,max,min,num); /\*输出该步计算过程\*/  
vr6174(num); /\*递归调用自身继续进行卡布列克运算\*/  
}  
}  
void parse\_sort(int num,int \*each)  
{  
int i,\*j,\*k,temp;  
for(i=0;i<=4;i++) /\*将NUM分解为数字\*/  
{  
j=each+3-i;  
\*j=num%10;  
num/=10;  
}  
for(i=0;i<3;i++) /\*对各保数字从小到大进行排序\*/  
for(j=each,k=each+1;j<each+3-i;j++,k++)  
if(\*j>\*k) { temp=\*j;\*j=\*k;\*k=temp;}  
return;  
}  
void max\_min(int \*each,int \*max,int \*min) /\*将分解的数字还原为最大整数和最小整数\*/  
{  
int \*i;  
\*min=0;  
for(i=each;i<each+4;i++) /\*还原为最小的整数\*/  
\*min=\*min\*10+\*i;  
\*max=0;  
for(i=each+3;i>=each;i–) /\*还原为最大的整数\*/  
\*max=\*max\*10+\*i;  
return;  
}

**\*运行结果**  
1) Enter a number:4312  
[1]:4312-1234=3078  
[2]:8730-378=8352  
[3]:8532-2358=6174  
2) Enter a number:8720  
[1]:8720-278=8442  
[2]:8442-2448=5994  
[3]:9954-4599=5355  
[4]:5553-3555=1998  
[5]:9981-1899=8082  
[6]:8820-288=8523  
[7]:8532-2358=6174  
3)Enter a number:9643  
[1]:9643-3469=6174

84.尼科彻斯定理

验证尼科彻斯定理，即：任何一个整数的立方都可以写成一串连续奇数的和。××

**\*问题分析与算法设计**  
本题是一个定理，我们先来证明它是成立的。  
对于任一正整数a,不论a是奇数还是偶数，整数(a×a-a+1)必然为奇数。  
构造一个等差数列，数列的首项为(a×a-a+1),等差数列的差值为2(奇数数列)，则前a项的和为：  
a×((a×a-a+1))+2×a(a-1)/2  
=a×a×a-a×a+a+a×a-a  
=a×a×a  
定理成立。证毕。  
通过定理的证明过程可知L所要求的奇数数列的首项为(a×a-a+1)，长度为a。编程的算法不需要特殊设计，可按照定理的证明过直接进行验证。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int a,b,c,d;  
printf("Please enter a number:");  
scanf("%d",&a); /\*输入整数\*/  
b=a\*a\*a; /\*求整数的三次方\*/  
printf("%d\*%d\*%d=%d=",a,a,a,b);  
for(d=0,c=0;c<a;c++) /\*输出数列，首项为a\*a-a+1,等差值为2\*/  
{  
d+=a\*a-a+1+c\*2; /\*求数列的前a项的和\*/  
printf(c?"+%d":"%d",a\*a-a+1+c\*2);  
}  
if(d==b)printf(" Y/n"); /\*若条件满足则输出“Y”\*/  
else printf(" N/n"); /\*否则输出“N”\*/  
}

**\*运行结果**  
1) Please enter a number:13  
13\*13\*13=2197=157+159+161+163+165+167+169+171+173+175+177+179+181 Y  
2) Please enter a number:14  
14\*14\*14=2744=183+185+187+189+191+193+195+197+199+201+203+205+207+209 Y

**\*思考题**  
本题的求解方法是先证明，在证明的过程中找到编程的算法，然后实现编程。实际上我们也可以不进行证明，直接使用编程中常用的试探方法来找出该数列，验证该定理。请读者自行设计算法。当然这样得到的数列可能与用定理方法得到的数列不一样。

85.回文数的形成

任取一个十进制整数，将其倒过来后与原来的整数相加，得到一个新的整数后重复以上步聚，则最终可得到一个回文数。请编程验证。

**\*问题分析与算法设计**  
回文数的这一形成规则目前还属于一个猜想，尚未得到数学上的证明。有些回文数要经历上百个步聚才能获得。这里通过编程验证。  
题目中给出的处理过程很清楚，算法不需要特殊设计。可按照题目的叙述直接进行验证。

**\*程序说明与注释**  
#include<stdio.h>  
#define MAX 2147483647  
long re(long int);  
int nonres(long int s);  
int main()  
{  
long int n,m;  
int count=0;  
printf("Please enetr a number optionaly:");  
scanf("%ld",&n);  
printf("The generation process of palindrome:/n");  
while(!nonres((m=re(n))+n)) /\*判断整数与其反序数相加后是否为回文数\*/  
{  
if(m+n>=MAX)  
{  
printf(" input error,break./n");  
break;  
}  
else  
{  
printf("[%d]:%ld+%ld=%ld/n",++count,n,m,m+n);  
n+=m;  
}  
}  
printf("[%d]:%ld+%ld=%ld/n",++count,n,m,m+n); /\*输出最后得到的回文数\*/  
printf("Here we reached the aim at last!/n");  
}  
long re(long int a) /\*求输入整数的反序数\*/  
{  
long int t;  
for(t=0;a>0;a/=10) /\*将整数反序\*/  
t=t\*10+a%10;  
return t;  
}  
int nonres(long int s) /\*判断给定的整数是否是回文数\*/  
{  
if(re(s)==s) return 1; /\*若是回文数则返回１\*/  
else return 0; /\*否则返回　０\*/  
}

 86.自动发牌

一副扑克有52张牌，打桥牌时应将牌分给四个人。请设计一个程序完成自动发牌的工作。要求：黑桃用S(Spaces)表示；红桃用H(Hearts)表示；方块用D(Diamonds)表示；梅花用C(Clubs)表示。

**\*问题分析与算法设计**  
按照打桥牌的规定，每人应当有13张牌。在人工发牌时，先进行洗牌，然后将洗好的牌按一定的顺序发给每一个人。为了便于计算机模拟，可将人工方式的发牌过程加以修改：先确定好发牌顺序：1、2、3、4；将52张牌顺序编号：黑桃2对应数字0，红桃2对应数字1，方块2对应数字2，梅花2对应数字3，黑桃3对应数字4，红桃3对应数字5，…然后从52 张牌中随机的为每个人抽牌。  
这里采用C语言库函数的随机函数，生成0到51之间的共52个随机数，以产生洗牌后发牌的效果。  
\*程序与程序注释

#include<stdlib.h>  
#include<stdio.h>  
int comp(const void \*j,const void \*i);  
void p(int b[],char n[]);

int main(void)  
{  
static char n[]={'2','3','4','5','6','7','8','9','T','J','Q','K','A'};  
int a[53],b1[13],b2[13],b3[13],b4[13];  
int b11=0,b22=0,b33=0,b44=0,t=1,m,flag,i;  
while(t<=52) /\*控制发52张牌\*/  
{  
m=rand()%52; /\*产生0到51之间的随机数\*/  
for(flag=1,i=1;i<=t&&flag;i++)/\*查找新产生的随机数是否已经存在\*/  
if(m==a[i]) flag=0; /\*flag=1:产生的是新的随机数flag=0:新产生的随机数已经存在\*/

if(flag)  
{  
a[t++]=m; /\*如果产生了新的随机数，则存入数组\*/  
if(t%4==0) b1[b11++]=a[t-1]; /\*根据t的模值，判断当前\*/  
else if(t%4==1) b2[b22++]=a[t-1]; /\*的牌应存入哪个数组中\*/  
else if(t%4==2) b3[b33++]=a[t-1];  
else if(t%4==3) b4[b44++]=a[t-1];  
}  
}

qsort(b1,13,sizeof(int),comp); /\*将每个人的牌进行排序\*/  
qsort(b2,13,sizeof(int),comp);  
qsort(b3,13,sizeof(int),comp);  
qsort(b4,13,sizeof(int),comp);  
p(b1,n); p(b2,n); p(b3,n); p(b4,n); /\*分别打印每个人的牌\*/

return 0;  
}

void p(int b[],char n[])  
{  
int i;  
printf("/n/006 "); /\*打印黑桃标记\*/  
for(i=0;i<13;i++) /\*将数组中的值转换为相应的花色\*/  
if(b[i]/13==0) printf("%c ",n[b[i]%13]); /\*该花色对应的牌\*/  
printf("/n/003 "); /\*打印红桃标记\*/

for(i=0;i<13;i++)  
if((b[i]/13)==1) printf("%c ",n[b[i]%13]);  
printf("/n/004 "); /\*打印方块标记\*/  
for(i=0;i<13;i++)  
if(b[i]/13==2) printf("%c ",n[b[i]%13]);  
printf("/n/005 "); /\*打印梅花标记\*/  
for(i=0;i<13;i++)  
if(b[i]/13==3||b[i]/13==4) printf("%c ",n[b[i]%13]);  
printf("/n");  
}

int comp(const void \*j,const void \*i) /\*qsort调用的排序函数\*/  
{  
return(\*(int\*)i-\*(int\*)j);  
}

87.黑白子交换

有三个白子和三个黑子如下图布置：  
○ ○ ○ . ● ● ●

游戏的目的是用最少的步数将上图中白子和黑子的位置进行交换：  
● ● ● . ○ ○ ○

游戏的规则是：(1)一次只能移动一个棋子； (2)棋子可以向空格中移动，也可以跳过一个对方的棋子进入空格，但不能向后跳，也不能跳过两个子。请用计算机实现上述游戏。

**\*问题分析与算法设计**  
计算机解决胜这类问题的关键是要找出问题的规律，或者说是要制定一套计算机行动的规则。分析本题，先用人来解决问题，可总结出以下规则：  
(1) 黑子向左跳过白子落入空格，转(5)  
(2) 白子向右跳过黑子落入空格，转(5)  
(3) 黑子向左移动一格落入空格(但不应产生棋子阻塞现象)，转(5)  
(4) 白子向右移动一格落入空格(但不应产生棋子阻塞现萌)，转(5)  
(5) 判断游戏是否结束，若没有结束，则转(1)继续。  
所谓的“阻塞”现象就是：在移动棋子的过程中，两个尚未到位的同色棋子连接在一起，使棋盘中的其它棋子无法继续移动。例如按下列方法移动棋子：  
0  
○ ○ ○ . ● ● ●   
1 ○ ○ . ○ ● ● ●   
2 △ ○ ○ ● ○ . ● ●   
3  
○ ○ ● . ○ ● ●   
4 两个●连在一起产生阻塞  
○ ○ ● ● ○ . ●   
或4 两个白连在一起产生阻塞  
○ . ● ○ ○ ● ●

产生阻塞的现象的原因是在第2步(△状态)时，棋子○不能向右移动，只能将●向左移动。  
总结产生阻塞的原因，当棋盘出现“黑、白、空、黑”或“白、空、黑、白”状态时，不能向左或向右移动中间的棋子，只移动两边的棋子。  
按照上述规则，可以保证在移动棋子的过程中，不会出现棋子无法移动的现象，且可以用最少的步数完成白子和黑子的位置交换。

**\*程序说明与注释**  
#include<stdio.h>  
int number;  
void print(int a[]);  
void change(int \*n,int \*m);  
int main()  
{  
int t[7]={1,1,1,0,2,2,2}; /\*初始化数组1：白子 2：黑子 0：空格\*/  
int i,flag;  
print(t);  
while(t[0]+t[1]+t[2]!=6||t[4]+t[5]+t[6]!=3) /\*判断游戏是否结束  
若还没有完成棋子的交换则继续进行循环\*/  
{  
flag=1; /\*flag 为棋子移动一步的标记1：尚未移动棋子 0：已经移动棋子\*/  
for(i=0;flag&&i<5;i++) /\*若白子可以向右跳过黑子，则白子向右跳\*/  
if(t[i]==1&&t[i+1]==2&&t[i+2]==0)  
{change(&t[i],&t[i+2]); print(t); flag=0;}  
for(i=0;flag&&i<5;i++) /\*若黑子可以向左跳过白子，则黑子向左跳\*/  
if(t[i]==0&&t[i+1]==1&&t[i+2]==2)  
{change(&t[i],&t[i+2]); print(t); flag=0;}  
for(i=0;flag&&i<6;i++) /\*若向右移动白子不会产生阻塞,则白子向右移动\*/  
if(t[i]==1&&t[i+1]==0&&(i==0||t[i-1]!=t[i+2]))  
{change(&t[i],&t[i+1]); print(t);flag=0;}  
for(i=0;flag&&i<6;i++) /\*若向左移动黑子不会产生阻塞，则黑子向左移动\*/  
if(t[i]==0&&t[i+1]==2&&(i==5||t[i-1]!=t[i+2]))  
{ change(&t[i],&t[i+1]); print(t);flag=0;}  
}  
}  
void print(int a[])  
{  
int i;  
printf("No. %2d:………………………../n",number++);  
printf(" ");  
for(i=0;i<=6;i++)  
printf(" | %c",a[i]==1?'\*':(a[i]==2?'@':' '));  
printf(" |/n ………………………../n/n");  
}  
void change(int \*n,int \*m)  
{  
int term;  
term=\*n; \*n=\*m; \*m=term;  
}

**\*问题的进一步讨论**  
本题中的规则不仅适用于三个棋子的情况，而且可以推而广之，适用于任意N个棋子的情况。读者可以编程验证，按照本规则得到的棋子移动步数是最少的。  
事实上，制定规则是解决这类问题的关键。一个游戏程序“思考水平的高低，完全取决于使用规则的好坏。”

**\*思考题**  
有两个白子和两个黑子如下左图布置：

○ . ○   
. . .   
● . ●

棋盘中的棋子按”马步“规则行走，要求用最少的步数将图中白子和黑子的位置进行交换，最终结果如下一幅图所示。  
● . ●   
. . .   
○ . ○

88.常胜将军

现有21根火柴，两人轮流取，每人每次可以取走1至4根，不可多取，也不能不取，谁取最后一楰火柴谁输。请编写一个程序进行人机对弈，要求人先取，计算机后取；计算机一方为“常胜将军”。

**\*问题分析与算法设计**  
在计算机后走的情况下，要想使计算机成为“常胜将军”，必须找出取 关键。根据本题的要求枷以总结出，后走一方取子的数量与对方刚才一步取子的数量之和等于，就可以保证最后一个子是留给先取子的那个人的。  
据此分析进行算法设计就是很简单的工作，编程实现也十分容易。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int a=21,i;  
printf("Game begin:/n");  
while(a>0)  
{  
do{  
printf("How many stick do you wish to take(1~%d)?",a>4?4:a);  
scanf("%d",&i);  
}while(i>4||i<1||i>a); /\*接收正在确的输入\*/  
if(a-i>0) printf(" %d stick left in the pile./n",a-i);  
if((a-i)<=0)  
{  
printf(" You have taken the last stick./n");  
printf(" \* \* \* You lose! /nGame Over./n"); /\*输出取胜标记\*/  
break;  
}  
else  
printf(" Compute take %d stick./n",5-i); /\*输出计算机取的子数\*/  
a-=5;  
printf(" %d stick left in the pile./n",a);  
}  
}

**\*思考题**  
改变题目中火柴的数量(如为22根)，则后走的一方就不一定能够保持常胜了，很可能改变成“常败”。此时后走一方的胜负就与火柴的初始数量和每次允许取的火柴数量的最大值有直接关系，请编写程序解决这一问题。

89.抢30

这是中国民间的一个游戏。两人从1开始轮流报数，每人每次可报一个数或两个连续的数，谁先报到30，谁就为胜方。

**\*问题分析与算法设计**  
本题与上题类似，算法也类似，所不同的是，本谁先走第一步是可选的。若计算机走第一步，那么计算机一定是赢家。若人先走一步，那么计算机只好等待人犯错误，如果人先走第一步且不犯错误，那么人就会取胜；否则计算机会抓住人的一次错误使自己成为胜利者。

**\*程序说明与注释**  
#include<stdio.h>  
#include<time.h>  
#include<stdlib.h>  
int input(int t);  
int copu(int s);  
int main()  
{  
int tol=0;  
printf("/n\* \* \* \* \* \* \* \*catch thirty\* \* \* \* \* \* \* /n");  
printf("Game Begin/n");  
randomize(); /\*初始化随机数发生器\*/  
if(random(2)==1) /\*取随机数决定机器和人谁先走第一步\*/  
tol=input(tol); /\*若为1，则余元走第一步\*/  
while(tol!=30) /\*游戏结束条件\*/  
if((tol=copu(tol))==30) /\*计算机取一个数，若为30则机器胜利\*/  
printf("I lose! /n");  
else  
if((tol=input(tol))==30) /\*人取一个数，若为30则人胜利\*/  
printf("I lose! /n");  
printf(" \* \* \* \* \* \* \* \*Game Over \* \* \* \* \* \* \* \*/n");  
}  
int input(int t)  
{  
int a;  
do{  
printf("Please count:");  
scanf("%d",&a);  
if(a>2||a<1||t+a>30)  
printf("Error input,again!");  
else  
printf("You count:%d/n",t+a);  
}while(a>2||a<1||t+a>30);  
return t+a; /\*返回当前的已经取走的数累加和\*/  
}  
int copu(int s)  
{  
int c;  
printf("Computer count:");  
if((s+1)%3==0) /\*若剩余的数的模为1，则取1\*/  
printf(" %d/n",++s);  
else if((s+2)%3==0)  
{  
s+=2; /\*若剩余的数的模为2，则取2\*/  
printf(" %d/n",s);  
}  
else  
{  
c=random(2)+1; /\*否则随机取1或2\*/  
s+=c;  
printf(" %d/n",s);  
}  
return s;  
}

**\*思考题**  
巧夺偶数。桌子上有25颗棋子，游戏双方轮流取子，每人每次最少取走一颗棋子，最多可取走3颗棋子。双方照这样取下去，直到取光所有的棋子。于是双方手中必然一方为偶数，一方为奇数，偶数方为胜者。请编程实现人机游戏。

90.搬山游戏

设有n座山，计算机与人为比赛的双方，轮流搬山。规定每次搬山的数止不能超 过k座，谁搬最后一座谁输。游戏开始时。计算机请人输入山的总数(n)和每次允许搬山的最大数止(k)。然后请人开始，等人输入了需要搬走的山的数目后，计算机马上打印出它搬多少座山，并提示尚余多少座山。双方轮流搬山直到最后一座山搬完为止。计算机会显示谁是赢家，并问人是否要继续比赛。若人不想玩了，计算机便会统计出共玩了几局，双方胜负如何。

**\*问题分析与算法设计**  
计算机参加游戏时应遵循下列原则：  
1) 当：  
剩余山数目-1<=可移动的最大数k 时计算机要移(剩余山数目-1)座，以便将最后一座山留给人。  
2)对于任意正整数x,y，一定有：  
0<=x%(y+1)<=y  
在有n座山的情况下，计算机为了将最后一座山留给人，而且又要控制每次搬山的数目不超过最大数k，它应搬山的数目要满足下列关系：  
(n-1)%(k+1)  
如果算出结果为0，即整除无余数，则规定只搬1座山，以防止冒进后发生问题。  
按照这样的规律，可编写出游戏程序如下：  
#include<stdio.h>  
int main()  
{  
int n,k,x,y,cc,pc,g;  
printf("More Mountain Game/n");  
printf("Game Begin/n");  
pc=cc=0;  
g=1;  
for(;;)  
{  
printf("No.%2d game /n",g++);  
printf("—————————————/n");  
printf("How many mpuntains are there?");  
scanf("%d",&n);  
if(!n) break;  
printf("How many mountains are allowed to each time?");  
do{  
scanf("%d",&k);  
if(k>n||k<1) printf("Repeat again!/n");  
}while(k>n||k<1);  
do{  
printf("How many mountains do you wish movw away?");  
scanf("%d",&x);  
if(x<1||x>k||x>n) /\*判断搬山数是否符合要求\*/  
{  
printf("IIIegal,again please!/n");  
continue;  
}  
n-=x;  
printf("There are %d mountains left now./n",n);  
if(!n)  
{  
printf("……………I win. You are failure……………/n/n");cc++;  
}  
else  
{  
y=(n-1)%(k+1); /\*求出最佳搬山数\*/  
if(!y) y=1;  
n-=y;  
printf("Copmputer move %d mountains away./n",y);  
if(n) printf(" There are %d mountains left now./n",n);  
else  
{  
printf("……………I am failure. You win………………/n/n");  
pc++;  
}  
}  
}while(n);

}  
printf("Games in total have been played %d./n",cc+pc);  
printf("You score is win %d,lose %d./n",pc,cc);  
printf("My score is win %d,lose %d./n",cc,pc);  
}

**\*思考题**  
取石子游戏。将石子分成若干堆，每堆有若干粒，参加游戏的甲乙两方轮流从任意一堆中取走任意个石子，甚至可以全部取走，但每次只能在一堆中取，不允许从这堆取一些，再从另一堆中取一些。直到谁取走最后一粒石子谁就获胜。请编程进行人机对弈

91.人机猜数游戏

由计算机“想”一个四位数，请人猜这个四位数是多少。人输入四位数字后，计算机首先判断这四位数字中有几位是猜对了，并且在对的数字中又有几位位置也是对的，将结果显示出来，给人以提示，请人再猜，直到人猜出计算机所想的四位数是多少为止。  
例如：计算机“想”了一个“1234”请人猜，可能的提示如下：  
人猜的整数 计算机判断有几个数字正确 有几个位置正确  
1122 2 1  
3344 2 1  
3312 3 0  
4123 4 0  
1243 4 2  
1234 4 4  
游戏结束  
请编程实现该游戏。游戏结束时，显示人猜一个数用了几次。

**\*问题分析与算法设计**  
问题本身清楚明了。判断相同位置上的数字是否相同不需要特殊的算法。只要截取相同位置上的数字进行比较即可。但在判断几位数字正确时，则应当注意：计算机所想的是“1123”，而人所猜的是“1576”，则正确的数字只有1位。  
程序中截取计算机所想的数的每位数字与人所猜的数字按位比较。若有两位数字相同，则要记信所猜中数字的位置，使该位数字只能与一位对应的数字“相同”。当截取下一位数字进行比较时，就不应再与上述位置上的数字进行比较，以避免所猜的数中的一位与对应数中多位数字“相同”的错误情况。

**\*程序说明与注释**  
#include<stdio.h>  
#include<time.h>  
#include<stdlib.h>  
int main()  
{  
int stime,a,z,t,i,c,m,g,s,j,k,l[4]; /\*j:数字正确的位数 k:位置正确的位数\*/  
long ltime;  
ltime=time(NULL); /\*l:数字相同时，人所猜中数字的正确位置\*/  
stime=(unsigned int)ltime/2;  
srand(stime);  
z=random(9999); /\*计算机想一个随机数\*/  
printf("I have a number with 4 digits in mind,please guess./n");  
for(c=1;;c++) /\*c: 猜数次数计数器\*/  
{  
printf("Enter a number with 4 digits:");  
scanf("%d",&g); /\*请人猜\*/   
a=z;j=0;k=0;l[0]=l[1]=l[2]=l[3]=0;  
for(i=1;i<5;i++) /\*i:原数中的第i位数。个位为第一位，千位为第4位\*/  
{  
s=g;m=1;  
for(t=1;t<5;t++) /\*人所猜想的数\*/  
{  
if(a%10==s%10) /\*若第i位与人猜的第t位相同\*/  
{  
if(m&&t!=l[0]&&t!=l[1]&&t!=l[2]&&t!=l[3])  
{  
j++;m=0;l[j-1]=t; /\*若该位置上的数字尚未与其它数字“相同”\*/  
} /\*记录相同数字时，该数字在所猜数字中的位置\*/

if(i==t) k++; /\*若位置也相同，则计数器k加1\*/  
}  
s/=10;  
}  
a/=10;  
}  
printf("You hane correctly guessed %d digits,/n",j);  
printf("and correctly guessed %d digits in exact position./n",k);  
if(k==4) break; /\*若位置全部正确，则人猜对了，退出\*/  
}  
printf("Now you have correctly guessed the whole number after %d times./n",c);  
}

Now you have correctly guessed the whole number after 7 times.

**\*思考题**  
猜数游戏。由计算机“想”一个数请人猜，人输入猜的数，如果猜对了，则结束游戏，否则计算机会给出提示，指出人猜的数是太大，还是太小。当一个数猜了20次还未猜中时，应停止猜数者继续游戏的权力，从程序中退出。

 92.人机猜数游戏(2)

将以上游戏（[91.人机猜数游戏](http://null/10091.html)）双方倒一下，请人想一个四位的整数，计算机来猜，人给计算机提示信息，最终看计算机用几次猜出一个人“想”的数。请编程实现。

**\*问题分析与算法设计**  
解决这类问题时，计算机的思考过程不可能象人一样具完备的推理能力，关键在于要将推理和判断的过程变成一种机械的过程，找出相应的规则，否则计算机难以完成推理工作。  
基于对问题的分析和理解，将问题进行简化，求解分为两个步聚来完成：首先确定四位数字的组成，然后再确定四位数字的排列顺序。可以列出如下规则：  
1)分别显示四个1，四个2，……，四个0，确定四位数字的组成。  
2)依次产生四位数字的全部排列(依次两两交换全部数字的位置)。  
3)根据人输入的正确数字及正确位置的数目，进行分别处理：  
(注意此时不出现输入的情况，因为在四个数字已经确定的情况下，若有3个位置正确，则第四个数字的位置必然也是正确的)  
若输入4：游戏结束。  
判断本次输入与上次输入的差值  
若差为2：说明前一次输入的一定为0，本次输入的为2，本次交换的两个数字的位置是正确的，只要交换另外两个没有交换过的数字即可结束游戏。  
若差为-2：说明前一次输入的一定为2，本次的一定为0。说明刚交换过的两个数字的位置是错误的，只要将交换的两个数字位置还原，并交换另外两个没有交换过的数字即可结束游戏。  
否则：若本次输入的正确位置数<=上次的正确位置数  
则恢复上次四位数字的排列，控制转3)  
否则：将本次输入的正确位置数作为“上次输入的正确位置数”，控制转3)。

**\*程序说明与注释**  
#include<stdio.h>  
#include<stdlib.h>  
void bhdy(int s,int b);  
void prt();  
int a[4],flag,count;  
int main()  
{  
int b1,b2,i,j,k=0,p,c;  
printf("Game guess your number in mind is # # # #./n");  
for(i=1;i<10&&k<4;i++) /\*分别显示四个1~9确定四个数字的组成\*/  
{  
printf("No.%d:your number may be:%d%d%d%d/n",++count,i,i,i,i);  
printf("How many digits have bad correctly guessed:");  
scanf("%d",&p); /\*人输入包含几位数字\*/   
for(j=0;j<p;j++)  
a[k+j]=i; /\*a[]:存放已确定数字的数组\*/  
k+=p; /\*k:已确定的数字个数\*/  
}  
if(k<4) /\*自动算出四位中包的个数\*/  
for(j=k;j<4;j++)  
a[j]=0;  
i=0;  
printf("No.%d:your number may be:%d%d%d%d/n",++count,a[0],a[1],a[2],a[3]);  
printf("How many are in exact positions:"); /\*顺序显示四位数字\*/  
scanf("%d",&b1); /\*人输入有几位位置是正确的\*/  
if(b1==4){prt();exit(0);} /\*四位正确，打印结果。结束游戏\*/  
for(flag=1,j=0;j<3&&flag;j++) /\*实现四个数字的两两(a[j],a[k]交换\*/  
for(k=j+1;k<4&&flag;k++)  
if(a[j]!=a[k])  
{  
c=a[j];a[j]=a[k];a[k]=c; /\*交换a[j],a[k]\*/  
printf("No.%d:Your number may be: %d%d%d%d/n",++count,a[0],a[1],a[2],a[3]);  
printf("How many are in exact positins:");  
scanf("%d",&b2); /\*输入有几个位置正确\*/  
if(b2==4){prt();flag=0;} /\*若全部正确，结束游戏\*/  
else if(b2-b1==2)bhdy(j,k); /\*若上次与本次的差为2，则交换两个元素即可结束\*/  
else if(b2-b1==-2) /\*若上次与本次的差为-2，则说明已交换的(a[j],a[k])是错误的  
将(a[j],a[k]还原后，只要交换另外两个元素即可结束游戏\*/  
{  
c=a[j];a[j]=a[k];a[k]=c;  
bhdy(j,k);  
}  
else if(b2<=b1)  
{  
c=a[j];a[j]=a[k];a[k]=c; /\*恢复交换的两个数字\*/  
}  
else b1=b2; /\*其它情况则将新输入的位置信息作为上次的位置保存\*/  
}  
if(flag) printf("You input error!/n"); /\*交换结果仍没结果，只能是人输入的信息错误\*/  
}  
void prt() /\*打印结果，结束游戏\*/  
{  
printf("Now your number must be %d%d%d%d./n",a[0],a[1],a[2],a[3]);  
printf("Game Over/n");  
}  
void bhdy(int s,int b)  
{  
int i,c=0,d[2];  
for(i=0;i<4;i++) /\*查找s和b以外的两个元素下标\*/  
if(i!=s&&i!=b) d[c++]=i;  
i=a[d[1>;a[d[1>=a[d[0>; a[d[0>=i; /\*交换除a[s]和a[b]以外的两个元素\*/  
prt(); /\*打印结果，结束游戏\*/  
flag=0;  
}

\*运行示例  
假设人想的四位数是：7215  
Game Begin  
Now guess your number in mind is # # # #.  
No.1:your number may be:1111

**\*问题的进一步讨论**  
本程序具有逻辑结构清析、算法简单正确的优点，但在接受人的输入信息时缺少必要的出错保护功能，同时在进行第三步推理过程中没有保留每次猜出的数字位置信息及人输入的回答，这样对于每次人输入的信息就无法进行合法性检查，即无法检查人的输入信息是否自相矛盾；同晨也无法充分利用前面的结果。  
这些缺陷是可以改进的，但最后一个问题改进难度较大，留给大家自己去完成。

**\*思考题**  
“一条龙游戏”。在一个3×3的棋盘上，甲乙双方进行对弃，双方在棋盘上轮流放入棋子，如果一方的棋子成一直线(横、竖或斜线)，则该方赢。请编写该游戏程序实现人与机器的比赛。比赛结果有三种：输、赢或平。  
在编程过程中请首先分析比赛中怎样才能获胜，找出第一步走在什么位置就最可能赢

 93.汉诺塔

约19世纪末，在欧州的商店中出售一种智力玩具，在一块铜板上有三根杆，最左边的杆上自上而下、由小到大顺序串着由64个圆盘构成的塔。目的是将最左边杆上的盘全部移到右边的杆上，条件是一次只能移动一个盘，且不允许大盘放在小盘的上面。

**\*问题分析与算法设计**  
这是一个著名的问题，几乎所有的教材上都有这个问题。由于条件是一次只能移动一个盘，且不允许大盘放在小盘上面，所以64个盘的移动次数是：  
18，446，744，073，709，551，615  
这是一个天文数字，若每一微秒可能计算(并不输出)一次移动，那么也需要几乎一百万年。我们仅能找出问题的解决方法并解决较小N值时的汉诺塔，但很难用计算机解决64层的汉诺塔。  
分析问题，找出移动盘子的正确算法。  
首先考虑a杆下面的盘子而非杆上最上面的盘子，于是任务变成了：  
\*将上面的63个盘子移到b杆上；  
\*将a杆上剩下的盘子移到c杆上；  
\*将b杆上的全部盘子移到c杆上。  
将这个过程继续下去，就是要先完成移动63个盘子、62个盘子、61个盘子….的工作。  
为了更清楚地描述算法，可以定义一个函数movedisc(n,a,b,c)。该函数的功能是：将N个盘子从A杆上借助C杆移动到B杆上。这样移动N个盘子的工作就可以按照以下过程进行：  
1) movedisc(n-1,a,c,b);  
2) 将一个盘子从a移动到b上；  
3) movedisc(n-1,c,b,a)；  
重复以上过程，直到将全部的盘子移动到位时为止。

**\*程序说明与注释**  
#include<stdio.h>  
void movedisc(unsigned n,char fromneedle,char toneedle,char usingneedle);  
int i=0;  
int main()  
{  
unsigned n;  
printf("please enter the number of disc:");  
scanf("%d",&n); /\*输入N值\*/  
printf("/tneedle:/ta/t b/t c/n");  
movedisc(n,'a','c','b'); /\*从A上借助B将N个盘子移动到C上\*/  
printf("/t Total: %d/n",i);  
}  
void movedisc(unsigned n,char fromneedle,char toneedle,char usingneedle)  
{  
if(n>0)  
{  
movedisc(n-1,fromneedle,usingneedle,toneedle);  
/\*从fromneedle上借助toneedle将N-1个盘子移动到usingneedle上\*/  
++i;  
switch(fromneedle) /\*将fromneedle 上的一个盘子移到toneedle上\*/  
{  
case 'a': switch(toneedle)  
{  
case 'b': printf("/t[%d]:/t%2d………>%2d/n",i,n,n);  
break;  
case 'c': printf("/t[%d]:/t%2d……………>%2d/n",i,n,n);  
break;  
}  
break;  
case 'b': switch(toneedle)  
{  
case 'a': printf("/t[%d]:/t%2d<……………>%2d/n",i,n,n);  
break;  
case 'c': printf("/t[%d]:/t %2d……..>%2d/n",i,n,n);  
break;  
}  
break;  
case 'c': switch(toneedle)  
{  
case 'a': printf("/t[%d]:/t%2d<…………%2d/n",i,n,n);  
break;  
case 'b': printf("/t[%d]:/t%2d<……..%2d/n",i,n,n);  
break;  
}  
break;  
}  
movedisc(n-1,usingneedle,toneedle,fromneedle);  
/\*从usingneedle上借助fromneedle将N-1个盘子移动到toneedle上\*/  
}  
}

94.兎子产子

从前有一对长寿兎子，它们每一个月生一对兎子，新生的小兎子两个月就长大了，在第二个月的月底开始生它们的下一代小兎子，这样一代一代生下去，求解兎子增长数量的数列。

**\*问题分析与算法设计**  
问题可以抽象成下列数学公式：  
Un=Un-1+Un-2  
其中：  
n是项数(n>=3)。它就是著名的菲波那奇数列，该数列的前几为：1，1，2，3，5，8，13，21…  
菲波那奇数列在程序中可以用多种方法进行处理。按照其通项递推公式利用最基本的循环控制就可以实现题目的要求。

**\*程序说明与注释**  
#include<stdio.h>  
int main()  
{  
int n,i,un1,un2,un;  
for(n=2;n<3;)  
{  
printf("Please enter required number of generation:");  
scanf("%d",&n);  
if(n<3) printf("/n Enter error!/n"); /\*控制输入正确的N值\*/  
}  
un=un2=1;  
printf("The repid increase of rabbits in first %d generation is as felow:/n",n);  
printf("l/tl/t");  
for(i=3;i<=n;i++)  
{  
un1=un2;  
un2=un;  
un=un1+un2; /\*利用通项公式求解N项的值\*/  
printf(i%10?"%d/t":"%d/n",un);  
}  
printf("/n");  
}

**\*运行结果**  
Please enter required number of generation: 20  
The repid increase of rabbits in first 20 generation is as felow:  
1 1 2 3 5 8 13 21 34 55  
89 144 233 377 610 987 1597 2584 4181 6765

95.将阿拉伯数字转换为罗马数字

将大于0小于1000的阿拉伯数字转换为罗马数字。阿拉伯数字与罗马数字的对应关系如下：  
1 2 3 4 5 ……  
I II III IV V ……

**\*问题分析与算法设计**  
题目中给出了阿拉伯数字与罗马数字的对应关系，题中的数字转换实际上就是查表翻译。即将整数的百、十、个位依次从整数中分解出来，查找表中相应的行后输出对应的字符。  
\*程序与程序设计  
#include<stdio.h>  
int main()  
{  
static char \*a[][10]={"","I","II","III","IV","V","VI","VII","VIII","IX"  
"","X","XX","XXX","XL","L","LX","LXX","LXXX","XCC",  
"","C","CC","CCC","CD","D","DC","DCC","DCCC","CM"  
}; /\*建立对照表\*/  
int n,t,i,m;  
printf("Please enter number:");  
scanf("%d",&n); /\*输入整数\*/  
printf("%d=",n);  
for(m=0,i=1000;m<3;m++,i/=10)  
{  
t=(n%i)/(i/10); /\*从高位向低位依次取各位的数字\*/  
printf("%s",a[2-m][t]); /\*通过对照表翻译输出\*/  
}  
printf("/n");  
}

**\*运行结果**  
1. Please enter number:863  
863=DCCCLXIII  
2. Please enter number: 256  
256=CCLVI  
3. Please enter number:355  
355=CCCLV  
4. Please enter number:522  
522=DXXII  
5. Please enter number:15  
15=XV

**\*思考题**  
输入正整数N，产生对应的英文数字符串并输出，例如：  
1 ONE 2 TWO 3 THREE  
10 TEN 11 ELEVEN  
135 ONE HUNDRED THIRTY FIVE

96.选美比赛

在选美大奖赛的半决胜赛现场，有一批选手参加比赛，比赛的规则是最后得分越高，名次越低。当半决决赛结束时，要在现场按照选手的出场顺序宣布最后得分和最后名次，获得相同分数的选手具有相同的名次，名次连续编号，不用考虑同名次的选手人数。例如：  
选手序号： 1，2，3，4，5，6，7  
选手得分： 5，3，4，7，3，5，6  
则输出名次为： 3，1，2，5，1，3，4  
请编程帮助大奖赛组委会完成半决赛的评分和排名工作。

**\*问题分析与算法设计**  
问题用程序设计语言加以表达的话，即为：将数组A中的整数从小到大进行连续编号，要求不改变数组中元素的顺序，且相同的整数要具有相同的编号。  
普通的排序方法均要改变数组元素原来的顺序，显然不能满足要求。为此，引入一个专门存放名次的数组，再采用通常的算法：在尚未排出名次的元素中找出最小值，并对具有相同值的元素进行处理，重复这一过程，直到全部元素排好为止。

**\*程序说明与注释**  
#include<stdio.h>  
#define NUM 7 /\*定义要处理的人数\*/  
int a[NUM+1]={0,5,3,4,7,3,5,6}; /\*为简单直接定义选手的分数\*/  
int m[NUM+1],l[NUM+1]; /\*m:已编名次的标记数组 l:记录同名次元素的下标\*/  
int main()  
{  
int i,smallest,num,k,j;  
num=1; /\*名次\*/  
for(i=1;i<=NUM;i++) /\*控制扫描整个数组，每次处理一个名次\*/  
if(m[i]==0) /\*若尚未进行名次处理(即找到第一个尚未处理的元素)\*/  
{  
smallest=a[i]; /\*取第一个未处理的元素作为当前的最小值\*/  
k=1; /\*数组l的下标，同名次的人数\*/  
l[k]=i; /\*记录分值为smallest的同名次元素的下标\*/  
for(j=i+1;j<=NUM;j++) /\*从下一个元素开始对余下的元素进行处理\*/  
if(m[j]==0) /\*若为尚未进行处理的元素\*/  
if(a[j]<smallest) /\*分数小于当前最小值\*/  
{  
smallest=a[j]; /\*则重新设置当覵最小值\*/  
k=0; /\*重新设置同名次人数\*/  
l[++k]=j; /\*重新记录同名次元素下标\*/  
}  
else if(a[j]==smallest) /\*若与当前最低分相同\*/  
l[++k]=j; /\*记录同名次的元素下标\*/  
for(j=1;j<=k;j++) /\*对同名次的元素进行名次处理\*/  
m[l[j>=num;  
num++; /\*名次加1\*/  
i=0; /\*控制重新开始，找下一个没排名次的元素\*/  
}  
printf("Player-No score Rank/n");  
for(j=1;j<=NUM;j++) /\*控制输出\*/  
printf(" %3d %4d %4d/n",j,a[j],m[j]);  
}

**\*运行结果**  
Player-No Score Rank  
1 5 3  
2 3 1  
3 4 2  
5 7 5  
5 3 1  
3 5 3  
7 6 4

**\*思考题**  
若将原题中的“名次连续编号，不用考虑同名次的选手人数”，改为”根据同名次的选手人数对选手的名次进行编号“，那么应该怎样修改程序。

97.满足特异条件的数列

输入m和n(20>=m>=n>0)求出满足以下方程的正整数数列 i1,i2,…,in，使得：i1+i1+…+in=m，且i1>=i2…>=in。例如：  
当n=4, m=8时，将得到如下5 个数列：  
5 1 1 1 4 2 1 1 3 3 1 1 3 2 2 1 2 2 2 2

**\*问题分析与算法设计**  
可将原题抽象为：将M分解为N个整数，且N个整数的和为M，i1>=i2>=…>=in。分解整数的方法很低多，由于题目中有"i1>=i2>=…..>=in，提示我们可先确定最右边in元素的值为1，然后按照条件使前一个元素的值一定大于等于当前元素的值，不断地向前推就可以解决问题。下面的程序允许用户选定M和N，输出满足条件的所有数列。

**\*程序说明与注释**  
#include<stdio.h>  
#define NUM 10 /\*允许分解的最大元素数量\*/  
int i[NUM]; /\*记录分解出的数值的数组\*/  
int main()  
{  
int sum,n,total,k,flag,count=0;  
printf("Please enter requried terms(<=10):");  
scanf("%d",&n);  
printf(" their sum:");  
scanf("%d",&total);  
sum=0; /\*当前从后向前k个元素的和\*/  
k=n; /\*从后向前正在处理的元素下标\*/  
i[n]=1; /\*将最后一个元素的值置为1作为初始值\*/  
printf("There are following possible series:/n");  
while(1)  
{  
if(sum+i[k]<total) /\*若后k位的和小于指定的total\*/  
if(k<=1) /\*若正要处理的是第一个元素\*/  
{i[1]=total-sum;flag=1;} /\*则计算第一个元素的并置标记\*/  
else{  
sum+=i[k–];  
i[k]=i[k+1]; /\*置第k位的值后k-1\*/  
continue; /\*继续向前处理其它元素\*/  
}  
else if(sum+i[k]>total||k!=1) /\*若和已超过total或不是第一个元素\*/  
{ sum-=i[++k]; flag=0;} /\*k向后回退一个元素\*/  
else flag=1; /\*sum+i[k]=total&&k=1 则设置flag标记\*/  
if(flag)  
{  
printf("[%d]:",++count);  
for(flag=1;flag<=n;++flag)  
printf("%d",i[flag]);  
printf("/n");  
}  
if(++k>n) /\*k向后回退一个元素后判断是否已退出最后一个元素\*/  
break;  
sum-=i[k];  
i[k]++; /\*试验下一个分解\*/  
}  
}

**\*运行结果**  
Please enter requried terms(<=10):4  
their sum:8  
There are following possible series:  
[1]: 5111  
[2]: 4211  
[3]: 3311  
[4]: 3221  
[5]: 2222

98.八皇后问题

在一个8×8国际象棋盘上，有8个皇后，每个皇后占一格；要求皇后间不会出现相互“攻击”的现象，即不能有两个皇后处在同一行、同一列或同一对角线上。问共有多少种不同的方法。

**\*问题分析与算法设计**  
这是一个古老的具有代表性的问题，用计算机求解时的算法也很多，这里仅介绍一种。  
采用一维数组来进行处理。数组的下标i表示棋盘上的第i列，a[i]的值表示皇后在第i列所放的位置。如：a[1]=5，表示在棋盘的第一例的第五行放一个皇后。  
程序中首先假定a[1]=1，表示第一个皇后放在棋盘的第一列的第一行的位置上，然后试探第二列中皇后可能的位置，找到合适的位置后，再处理后续的各列，这样通过各列的反复试探，可以最终找出皇后的全部摆放方法。  
程序采用回溯法，算法的细节参看程序。

**\*程序说明与注释**  
#include<stdio.h>  
#define NUM 8 /\*定义数组的大小\*/  
int a[NUM+1];  
int main()  
{  
int i,k,flag,not\_finish=1,count=0;  
i=1; /\*正在处理的元素下标，表示前i-1个元素已符合要求，正在处理第i个元素\*/  
a[1]=1; /\*为数组的第一个元素赋初值\*/  
printf("The possible configuration of 8 queens are:/n");  
while(not\_finish) /\*not\_finish=1:处理尚未结束\*/  
{  
while(not\_finish&&i<=NUM) /\*处理尚未结束且还没处理到第NUM个元素\*/  
{  
for(flag=1,k=1;flag&&k<i;k++) /\*判断是否有多个皇后在同一行\*/  
if(a[k]==a[i])flag=0;  
for(k=1;flag&&k<i;k++) /\*判断是否有多个皇后在同一对角线\*/  
if((a[i]==a[k]-(k-i))||(a[i]==a[k]+(k-i))) flag=0;  
if(!flag) /\*若存在矛盾不满足要求，需要重新设置第i个元素\*/  
{  
if(a[i]==a[i-1]) /\*若a[i]的值已经经过一圈追上a[i-1]的值\*/  
{  
i–; /\*退回一步，重新试探处理前一个元素\*/  
if(i>1&&a[i]==NUM)  
a[i]=1; /\*当a[i]为NUM时将a[i]的值置1\*/  
else if(i==1&&a[i]==NUM)  
not\_finish=0; /\*当第一位的值达到NUM时结束\*/  
else a[i]++; /\*将a[i]的值取下一个值\*/  
}  
else if(a[i]==NUM) a[i]=1;  
else a[i]++; /\*将a[i]的值取下一个值\*/  
}  
else if(++i<=NUM)  
if(a[i-1]==NUM) a[i]=1; /\*若前一个元素的值为NUM则a[i]=1\*/  
else a[i]=a[i-1]+1; /\*否则元素的值为前一个元素的下一个值\*/  
}  
if(not\_finish)  
{  
++count;  
printf((count-1)%3?" [%2d]: ":" /n[%2d]: ",count);  
for(k=1;k<=NUM;k++) /\*输出结果\*/  
printf(" %d",a[k]);  
if(a[NUM-1]<NUM) a[NUM-1]++; /\*修改倒数第二位的值\*/  
else a[NUM-1]=1;  
i=NUM-1; /\*开始寻找下一个足条件的解\*/  
}  
}  
}

**\*思考题**  
一个8×8的国际象棋盘，共有64个格子。最多将五个皇后放入棋盘中，就可以控制整个的盘面，不论对方的棋子放哪一格中都会被吃掉。请编程

99.超长正整数的加法

请设计一个算法来完成两个超长正整数的加法。

**\*问题分析与算法设计**  
首先要设计一种数据结构来表示一个超长的正整数，然后才能够设计算法。  
首先我们采用一个带有表头结点的环形链来表示一个非负的超大整数，如果从低位开始为每 个数字编号，则第一位到第四位、第五位到第八位…的每四位组成的数字，依次放在链表的第一个、第二个、…结点中，不足4位的最高位存放在链表的最后一个结点中，表头结点的值规定为-1。例如：  
大整数“587890987654321”可用如下的带表头结点head的链表表示：

按照此数据结构，可以从两个表头结点开始，顺序依次对应相加，求出所需要的进位后代入下面的运算。具体的实现算法请见程序中的注释。

**\*程序说明与注释**  
#include<stdio.h>  
#include<stdlib.h>  
#define HUNTHOU 10000  
typedef struct node{ int data;  
struct node \*next;  
}NODE; /\*定义链表结构\*/

NODE \*insert\_after(NODE \*u,int num); /\*在u结点后插入一个新的NODE，其值为num\*/  
NODE \*addint(NODE \*p,NODE \*q); /\*完成加法操作返回指向\*p+\*q结果的指针\*/  
void printint(NODE \*s);  
NODE \*inputint(void);

int main()  
{  
NODE \*s1,\*s2,\*s;  
NODE \*inputint(), \*addint(), \*insert\_after();  
printf("Enter S1= ");  
s1=inputint(); /\*输入被加数\*/  
printf("Enter S2= ");  
s2=inputint(); /\*输入加数\*/  
printf(" S1="); printint(s1); putchar('/n'); /\*显示被加数\*/  
printf(" S2="); printint(s2); putchar('/n'); /\*显示加数\*/  
s=addint(s1,s2); /\*求和\*/  
printf("S1+S2="); printint(s); putchar('/n'); /\*输出结果\*/  
}  
NODE \*insert\_after(NODE \*u,int num)  
{  
NODE \*v;  
v=(NODE \*)malloc(sizeof(NODE)); /\*申请一个NODE\*/  
v->data=num; /\*赋值\*/  
u->next=v; /\*在u结点后插入一个NODE\*/  
return v;  
}  
NODE \*addint(NODE \*p,NODE \*q) /\*完成加法操作返回指向\*p+\*q结果的指针\*/  
{  
NODE \*pp,\*qq,\*r,\*s,\*t;  
int total,number,carry;  
pp=p->next; qq=q->next;  
s=(NODE \*)malloc(sizeof(NODE)); /\*建立存放和的链表表头\*/  
s->data=-1;  
t=s; carry=0; /\*carry:进位\*/  
while(pp->data!=-1&&qq->data!=-1) /\*均不是表头\*/  
{  
total=pp->data+qq->data+carry; /\*对应位与前次的进位求和\*/  
number=total%HUNTHOU; /\*求出存入链中部分的数值 \*/  
carry=total/HUNTHOU; /\*算出进位\*/  
t=insert\_after(t,number); /\*将部分和存入s向的链中\*/  
pp=pp->next; /\*分别取后面的加数\*/  
qq=qq->next;  
}  
r=(pp->data!=-1)?pp:qq; /\*取尚未自理完毕的链指针\*/  
while(r->data!=-1) /\*处理加数中较大的数\*/  
{  
total=r->data+carry; /\*与进位相加\*/  
number=total%HUNTHOU; /\*求出存入链中部分的数值\*/  
carry=total/HUNTHOU; /\*算出进位\*/  
t=insert\_after(t,number); /\*将部分和存入s指向的链中\*/  
r=r->next; /\*取后面的值\*/  
}  
if(carry) t=insert\_after(t,1); /\*处理最后一次进位\*/  
t->next=s; /\*完成和的链表\*/  
return s; /\*返回指向和的结构指针\*/  
}  
NODE \*inputint(void) /\*输入超长正整数\*/  
{  
NODE \*s,\*ps,\*qs;  
struct number {int num;  
struct number \*np;  
}\*p,\*q;  
int i,j,k;  
long sum;  
char c;  
p=NULL; /\*指向输入的整数，链道为整数的最低的个位，链尾为整数的最高位\*/  
while((c=getchar())!='/n') /\*输入整数，按字符接收数字\*/  
if(c>='0'&&c<='9') /\*若为数字则存入\*/  
{  
q=(struct number \*)malloc(sizeof(struct number)); /\*申请空间\*/  
q->num=c-'0'; /\*存入一位整数\*/  
q->np=p; /\*建立指针\*/  
p=q;  
}  
s=(NODE \*)malloc(sizeof(NODE));  
s->data=-1; /\*建立表求超长正整数的链头\*/  
ps=s;  
while(p!=NULL) /\*将接收的临时数据链中的数据转换为所要求的标准形式\*/  
{  
sum=0;i=0;k=1;  
while(i<4&&p!=NULL) /\*取出低四位\*/  
{  
sum=sum+k\*(p->num);   
i++; p=p->np; k=k\*10;  
}  
qs=(NODE \*)malloc(sizeof(NODE)); /\*申请空间\*/  
qs->data=sum; /\*赋值，建立链表\*/  
ps->next=qs;  
ps=qs;  
}  
ps->next=s;  
return s;  
}  
void printint(NODE \*s)  
{  
if(s->next->data!=-1) /\*若不是表头，则输出\*/  
{  
printint(s->next); /\*递归输出\*/  
if(s->next->next->data==-1)  
printf("%d",s->next->data);  
else{  
int i,k=HUNTHOU;  
for(i=1;i<=4;i++,k/=10)  
putchar('0'+s->next->data%(k)/(k/10));  
}  
}  
}

**\*运行结果**

**\*思考题**

100.数字移动

在图中的九个点上,空出中间的点,其余的点上任意填入数字1到8;1的位置固定不动,然后移动其余的数字,使1到8顺时针从小到大排列.移动的规律是:只能将数字沿线移向空白的点.  
请编程显示数字移动过程。

**\*问题分析与算法设计**  
分析题目中的条件,要求利用中间的空白格将数字顺时针方向排列,且排列过程中只能借空白的点来移动数字.问题的实质就是将矩阵外面的8个格看成一个环,8个数字在环内进行排序,同于受题目要求的限制"只能将数字沿线移向空白的点",所以要利用中间的空格进行排序,这样要求的排序算法与众不同.  
观察中间的点,它是唯一一个与其它8个点有连线的点,即它是中心点.中心点的活动的空间最大,它可以向8个方向移动,充分利用中心点这个特性是算法设计成功与否的关键.  
在找到1所在的位置后,其余各个数字的正确位置就是固定的.我们可以按照下列算法从数字2开始,一个一个地来调整各个数字的位置.  
\*确定数字i应处的位置;  
\*从数字i应处的位置开始,向后查找数字i现在的位置;  
\*若数字i现在位置不正确,则将数字i从现在的位置(沿连线)移向中间的空格,而将原有位置空出;依次将现有空格前的所有元素向后移动;直到将i应处的位置空出,把它移入再次空出中间的格.  
从数字2开始使用以上过程,就可以完成全部数字的移动排序.  
编程时要将矩阵的外边八个格看成一个环,且环的首元素是不定的,如果算法设计得不好,程序中就要花很多精力来处理环中元素的前后顺序问题.将题目中的3X3矩阵用一个一维数组表示,中间的元素(第四号)刚好为空格,设计另一个指针数组,专门记录指针外八个格构成环时的连接关系.指针数组的每个元素依次记录环中数字在原来数组中对应的元素下标.这样通过指针数组将原来矩阵中复杂的环型关系表示成了简单的线性关系,从而大大地简化了程序设计.

**\*程序说明与注释**  
#include<stdio.h>  
int a[]={0,1,2,5,8,7,6,3}; /\*指针数组.依次存入矩阵中构成环的元素下标\*/  
int b[9]; /\*表示3X3矩阵,b[4]为空格\*/  
int c[9]; /\*确定1所在的位置后,对环进行调整的指针数组\*/  
int count=0; /\*数字移动步数计数器\*/  
int main()  
{   
int i,j,k,t;  
void print();  
printf("Please enter original order of digits 1~8:");  
for(i=0;i<8;i++)  
scanf("%d",&b[a[i>);  
/\*顺序输入矩阵外边的8个数字,矩阵元素的顺序由指针数组的元素a[i]控制\*/  
printf("The sorting process is as felow:/n");  
print();  
for(t=-1,j=0;j<8&&t==-1;j++) /\*确定数字1所在的位置\*/  
if(b[a[j>==1) t=j; /\*t:记录数字1所在的位置\*/   
for(j=0;j<8;j++) /\*调整环的指针数组,将数字1所在的位置定为环的首\*/  
c[j]=a[(j+t)%8];  
for(i=2;i<9;i++) /\*从2开始依次调整数字的位置\*/  
/\*i:正在处理的数字,i对应在环中应当的正确位置就是i-1\*/  
for(j=i-1;j<8;j++) /\*从i应处的正确位置开始顺序查找\*/  
if(b[c[j>==i&&j!=i-1) /\*若i不在正确的位置\*/  
{  
b[4]=i; /\*将i移到中心的空格中\*/  
b[c[j>=0;print(); /\*空出i原来所在的位置,输出\*/  
for(k=j;k!=i-1;k–) /\*将空格以前到i的正确位置之间的数字依次向后移动一格\*/  
{  
b[c[k>=b[c[k-1>; /\*数字向后移动\*/  
b[c[k-1>=0;  
print();  
}  
b[c[k>=i; /\*将中间的数字i移入正确的位置\*/  
b[4]=0; /\*空出中间的空格\*/  
print();  
break;  
}  
else if(b[c[j>==i) break; /\*数字i在正确的位置\*/  
}  
void print(void) /\*按格式要求输出矩阵\*/  
{  
int c;  
for(c=0;c<9;c++)  
if(c%3==2) printf("%2d ",b[c]);  
else printf("%2d",b[c]);  
printf("—-%2d—-/n",count++);  
}

**\*运行结果**

**\*问题的进一步讨论**  
很显然,按照上述算法都能解决问题，但移动的步数并不是最少的。  
注意算法中的两个问题。其一：数字1的位置自始自终是保持不变的；其2：没有考虑到初始情况下，位置原本就已经是正确的数字。如例中的数字5和6，按照算法，当移动其它数字时，5和6了要跟着移动多次，这显然费了不少步数。  
对于实例，若让数字1参与其它数字的移动排序过程，并充分利用数字5和6初始位置已经正确这一条件，可以大大优化移动排序的过程。

**\*思考题**  
请重新设计算法，编写更优化的程序，尽可能减少移动的步数。

请设计完成两个超长正整数的减法、乘法和除法的运算