# 计概期末超级题库答案

By 徐旺达

1.导弹拦截
2. 最长上升子序列
3.石子归并
4.三角形最佳路径问题（习题 15-9）
5.滑雪
6.鸡蛋的硬度（习题 15-12）
7.放苹果问题（习题 15-11）
8.最大子矩阵（习题 15-14）
9.采药（习题 15-13）
10.前缀表达式（逆波兰表达式）（习题 15-1）
11.大整数乘法（多组）
12.大整数除法（除以 13 类似）
13.大整数减法
14.阶乘的精确值
15.归并排序（分治）
16.快速排序
17.求排列的逆序数（分治归并）（习题 15-4）
18.多边形游戏
19.汉诺塔
20.汉诺塔移到中间那根柱子
21.八皇后问题（输出棋盘）
22.八皇后问题（输入数字）
23.试剂稀释（习题 15-10）
24.五户共井问题（习题 15-5）
25.称硬币
26.骑士游历
27.母牛繁殖问题
28.特殊日历计算（习题 10-6）
29.玛雅历（习题 10-7）
30.不吉利日期（习题 10-3）
31.日历问题（习题 10-1）
32.计算两个日期之间的天数（习题 10-2）
33.校门外的树（习题 11-2）
34.开关灯
35.删除数组中的元素（习题 11-4）
36.最匹配的矩阵（习题 11-7）
37.矩阵归零消减序列和（习题 11-8）
38.单词替换
39.数制转换（2 到 36 进制）
40.行程长度编码（习题 12-5）
41.回文子串（习题 12-6）
42.478-3279（习题 12-9）
43.选择你喜爱的水果（12-10）
44.计算 2 的 N 次方（习题 12-12）
45.n-gram 串频统计（习题 12-7）
46.电池的寿命（习题 13-3）
47.找第一个只出现一次的字符（习题 13-8）
48.第二个重复出现的数（习题 13-10）
49.挑选序列（习题 13-11）
50.白细胞计数（习题 13-12）
51.受限时间内的最小通行费（习题 13-13）
52.字符排序（习题 14-2）
53.距离排序（习题 14-4）
54.输出前 k 大的数（习题 14-5）
55.区间合并（习题 14-8）
56.找和为 k 的两个元素（习题 14-9）
57.矩阵乘法（习题 6-2）
58.二维数组回形遍历（习题 6-3）
59.二维数组右上左下遍历（习题 6-4）
60.矩阵交换行（习题 6-5）
61.哥德巴赫猜想
62.输出前 n 个素数（优化）
63.社会名流
64.蛇形填充数组
65.最大质因子序列（习题 9-1）
66.整数的质因子（习题 9-2）
67.因子分解（习题 9-3）
68.区间内的真素数（习题 9-4）
69.分解整数为质因数连乘
70.最小公倍数（习题 9-8）
71.约瑟夫问题
72.田忌赛马
73.弦截法求方程的根
74.填词
75.棋盘上的距离
76.装箱问题
77.密码（Bob and Alice）
78. skew 数
79.1082 子串
80.POJ 2804 词典
81.最短前缀
82.花生问题（多组）
83.排列
84.循环数（Round and Round We Go）
85.麦森数
86.浮点数加法
87.浮点数高精度幂
88.硬币面值
89.走出迷宫（习题 15-6）
90.城堡（习题 15-7）
91.生理周期
92.熄灯问题
93.恼人的青蛙
94.1160 建邮局 山区建小学（习题 15-8）
95.计算对数
96.数字方格
97. Doomsday
98. Saturday
99.分解因数
100.红与黑
101.棋盘问题
102.帮助吉米
103.最长公共子序列
104.陪审团的人选（copy）
105.矩形覆盖
106.金银岛
107. 1097 Pell 数列
108.木材分割
109.All in all
110.五子棋判输赢
111.最大子序列和
112.炮兵阵地

113.碱基的忧伤--计算机系男生的故事之一
114.括号匹配问题
115.最多点数直线问题
116.最简区间
117.三值排序
118.（附加题）公交线路
119.POJ1017 填充箱子
120.POJ1328 雷达装置
121.POJ1019 数字序列
122.POJ1154 字母
123.集合选数
124.判断是否为 C 语言的合法标识符
125.称体重
126.点评赛车
127.POJ2282 计数问题
128.POJ3974 求最长回文子串
129.涂色
130.子串替换
131.出现次数最多的字母
132.单词排序
133.求序列中的众数
134.1 的个数
135.站队
136.数字谜
137.配对碱基链
138.全排列
139.1049 跳绳游戏
140. 1081 字符串判等
141.拨钟问题
142.矩阵乘方和
143.计算并输出杨辉三角形的前 n 行
144.能种金币的聚宝盆
145.统计单词
146.神奇的口袋(2)
147.古代密码
148.平板上色
149.符号三角形
150.判断四边形
151.最大乘积
152.活动选择问题
153.旅行售货商问题（回溯）
154.象棋比赛（dp）
155.基因串
156.最小覆盖（贪心）
157.圆柱体装箱（贪心）
158.通信系统
159.积木正方形
160.字符游戏
161.翻转游戏
162.cleaning robot
163.布尔表达式
164.POJ1700 过河问题
165.取石子游戏
166.POJ1067 取石子游戏
167.POJ1821 粉刷篱笆（dp）
168.修复牛棚
169.连续邮资问题(回溯)
170.跳格问题
171.木棍正方形
172.POJ1681 画家问题
173.木棒分割（贪心）
174.佳佳的筷子（dp）
175.球桌出租（dp）
176.天平（dp）
177.路径计数
178.爆竹与箱子（dp）
179.合唱队形（dp）
180.能量项链（dp）
181.碎纸机
182.文件结构图
183.方格取数
184.过河
185.装箱问题（dp）
186.poj1159Palindrome
187. poj 1141 Brackets Sequence 括号匹配
188. A decorative fence
189. The Cow Lexicon 奶牛词典
190.dividing 邮票分配
191.AGTC
192.生日相同
193.正整数的任意进制转换（大整数）
194.最长等差数列子集
195.POJ2192Zipper
196.找次大数
197.两倍
198.循环移动
199.字符串排序
200.DNA 排序
201.和为 n 连续正数序列
202.最长最短单词
203.提取数字串按数值排序
204.降序生成进制数
205.木棒
206.字符串重排列
207.计算矩阵边缘元素之和
208.中间值判断
209. 判断整数的奇偶性
210.最简真分数序列
211.1102 迷宫
212.字符串最大跨距
213.啤酒厂选址
214.柱状图上的最大矩形
215.最大零矩阵

1.导弹拦截

某国为了防御敌国的导弹袭击，开发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，并观测到导弹依次飞来的高度，请计算这套系统最多能拦截多少导弹。拦截来袭导弹时，必须按来袭导弹袭击的时间顺序，不允许先拦截后面的导弹，再拦截前面的导弹。

分析：拦截第 i 枚导弹之后还最多能再拦截几枚导弹？从 i+1 到 k 循环，选择之后最大的枚数，加 1 即为拦截第 i 枚导弹之后还最多能拦截导弹的数目。最后从 1 到 k 循环，选择最大值，加上 1 便是最大拦截数。

```
#include<stdio.h>
int dp[100];
int main(void)
{
    int i,j,k;
    int m=0;
    int max=0;
    int high[100];
    scanf("%d",&k);
    for(i=1;i<=k;i++)
        scanf("%d",&high[i]);
    dp[k]=0;                                    //拦截完第 k 枚之后还能再打 0 颗
    for(i=k-1;i>=1;i--)
    {
        max=0;
        for(j=i+1;j<=k;j++)          //拦截完第 i 枚炮弹之后，还能在拦截几枚？只要在其后
    的所有 dp[j]+1 中取最大值，当然必须高度满足要求
        {
            if(high[i]>=high[j])
            {
                if(dp[j]+1>max)
                    max=dp[j]+1;
            }
        }
        dp[i]=max;
    }
    for(i=1;i<=k;i++)
    {
        if(dp[i]>m)                             //所有的 dp 中选择最大的。就从那一枚开始打。
            m=dp[i];
    }
    printf("%d",m+1);
    return 0;
}
```

2.最长上升子序列

一个数的序列 bi，当 b1 < b2 < ... < bS 的时候，我们称这个序列是上升的。对于给定的一个序列(a1, a2, ..., aN)，我们可以得到一些上升的子序列(ai1, ai2, ..., aiK)，这里 1 <= i1 < i2 < ... < iK <= N。比如，对于序列(1, 7, 3, 5, 9, 4, 8)，有它的一些上升子序列，如(1, 7), (3, 4, 8)等等。这些子序列中最长的长度是 4，比如子序列(1, 3, 5, 8).

分析：第 i 个数之后最长的上升子序列个数为？从 i+1 到 n 循环，选择最长的个数，加 1 即为第 i 个数之后最长的上升子序列个数。最后从 1 到 n 循环，选择最大的个数，加 1，即为最长子序列长度。

```
#include<stdio.h>
int dp[10000];
int num[10000];
int main(void)
{
    int i,j,k;
    int n;
    int max;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        scanf("%d",&num[i]);
    dp[n]=0;
    for(i=n-1;i>=1;i--)
```

```
    {
        max=0;
        for(j=i+1;j<=n;j++)
        {
            if(dp[j]+1>max&&num[i]<num[j])
                max=dp[j]+1;
        }
        dp[i]=max;
    }
    int m=0;
    for(i=1;i<=n;i++)
    {
        if(dp[i]>m)
            m=dp[i];
    }
    printf("%d",m+1);
    return 0;
}
```

3.石子归并

现摆一排 N 堆石子（N ≤ 100），要将石子有次序地合并成一堆。规定每次只能选取相邻的两堆合并成新的一堆，并将新的一堆的石子数，记为该次合并的得分。编一程序，由文件读入堆数 N 及每堆的石子数（ ≤ 20）。选择一种合并石子的方案，使所做 N－1 次合并，得分的总和最小。

分析：令 score[i][j]表示第 i 堆石子到第 j 堆石子合成一堆时候的总分，score[i][i]=0。当 i≠j 时，score[i][j]=min{score[i][k]+score[k+1][j]+sum[i][j]}，sum[i][j]表示第 i 堆到第 j 堆石子总数，k 从 i 取到 j。从间隔为 1 开始，一直到间隔为 n-1，最后只要输出 score[1][n]即为最小总和。

```
#include<stdio.h>
int score[101][101];
int number[101];
int f(int i,int j)
{
    int c=0,k;
    for(k=i;k<=j;k++)
        c+=number[k];
    return c;
}
int main(void)
{
    int i,j,k;
    int n;
    int min;
    int total=0;
    int minn;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        scanf("%d",&number[i]);
    for(i=1;i<=n;i++)
        score[i][i]=0;
    for(i=1;i<=n-1;i++)
    {
        for(j=1;j+i<=n;j++)
        {
            min=10000;
            for(k=j;k<=j+i-1;k++)
            {
                minn=score[j][k]+score[k+1][j+i]+f(j,j+i);
                if(minn<min)
```

```
                    min=minn;
                }
                score[j][j+i]=min;
            }
        }
        printf("%d",score[1][n]);
        return 0;
}
```

如是环形的，只要注意圆形问题转线形的方法：将石子序列倍长，答案在 s[i][i + n - 1]  (i = 0 .. n - 1)中找。

```
#include<stdio.h>
int score[202][202];
int number[202];
int f(int i,int j)
{
        int c=0,k;
        for(k=i;k<=j;k++)
                c+=number[k];
        return c;
}
int main(void)
{
        int i,j,k,x;
        int n;
        int min;
        int minn;
        scanf("%d",&n);
        for(i=1;i<=n;i++)
                scanf("%d",&number[i]);
        for(i=n+1;i<2*n;i++)
                number[i]=number[i-n];
        for(i=1;i<2*n;i++)
                score[i][i]=0;
        for(x=1;x<=n;x++)
        {
                for(i=1;i<=n-1;i++)
                {
                        for(j=x;j+i<=n+x-1;j++)
                        {
                                min=100000;
                                for(k=j;k<=j+i-1;k++)
                                {
                                        minn=score[j][k]+score[k+1][j+i]+f(j,j+i);
                                        if(minn<min)
                                                min=minn;
                                }
                                score[j][j+i]=min;
                        }
                }
        }
        min=1000000;
        for(i=1;i<=n;i++)
        {
                if(min>score[i][i+n-1])
                        min=score[i][i+n-1];
        }
        printf("%d",min);
        return 0;
```

}
4.三角形最佳路径问题
如下所示的由正整数数字构成的三角形:
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
从三角形的顶部到底部有很多条不同的路径。对于每条路径，把路径上面的数加起来可以得到一个和，和最大的路径称为最佳路径。你的任务就是求出最佳路径上的数字之和。
注意：路径上的每一步只能从一个数走到下一层上和它最近的下边（正下方）的数或者右边（右下方）的数。
分析：a[i][j]表示第 i 层第 j 个的长度，dp[i][j]表示以第 i 层第 j 个点为顶点，它的下面最长的路径是多少。从最下端开始，直到 dp[1][1]。

```
#include<stdio.h>
int a[101][101];
int dp[101][101];
int max(int x,int y)
{
	if(x>=y)
		return x;
	else
		return y;
}
int main(void)
{
	int i,j,k;
	int n;
	scanf("%d",&n);
	for(i=1;i<=n;i++)
		for(j=1;j<=i;j++)
			scanf("%d",&a[i][j]);
	for(i=1;i<=n;i++)
		dp[n][i]=a[n][i];
	for(i=n-1;i>=1;i--)
	{
		for(j=1;j<=i;j++)
			dp[i][j]=max(dp[i+1][j]+a[i][j],dp[i+1][j+1]+a[i][j]);
	}
	printf("%d",dp[1][1]);
	return 0;
}
```

5.滑雪
Michael 想知道载一个区域中最长的滑坡。区域由一个二维数组给出。数组的每个数字代表点的高度。下面是一个例子
1 2 3 4 5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
13 12 11 10 9
一个人可以从某个点滑向上下左右相邻四个点之一，当且仅当高度减小。在上面的例子中，一条可滑行的滑坡为 24-17-16-1。当然 25-24-23-...-3-2-1 更长。事实上，这是最长的一条。
输入的第一行表示区域的行数 R 和列数 C(1 <= R,C <= 100)。下面是 R 行，每行有 C 个整数，代表高度 h，0<=h<=10000。输出最长区域的长度。
分析：dp[i][j]以第 i 排第 j 列为顶点，所对应的最大滑行长度。f(i,j)表示给定一个高度后，向它的上下左右检索，如果高度比相邻高度高，且相邻的在界内，则此点的距离=相邻点距离+1,然后比较这所有的相邻点+1 的值，取最大值。之前 dp 已经初始化为 0 了，当碰到 dp[i][j]

≠0 的时候直接返回，因为这个点已经算过了，无需重复运算。这是动态规划与一般递归运算的差别，对已经算过的点进行记录，免去重复计算。

```
#include<stdio.h>
#include<string.h>
int dp[101][101];
int high[101][101];
int r,c;
int f(int i,int j);
int main(void)
{
	int i,j,k;
	int max=1;
	scanf("%d%d",&r,&c);
	for(i=1;i<=r;i++)
	{
		for(j=1;j<=c;j++)
		{
			scanf("%d",&high[i][j]);
		}
	}
	memset(dp,0,sizeof(dp));
	for(i=1;i<=r;i++)
	{
		for(j=1;j<=c;j++)
		{
			dp[i][j]=f(i,j);
			if(dp[i][j]>max)
				max=dp[i][j];
		}
	}
	printf("%d",max);
	return 0;
}
int f(int i,int j)
{
	int lenmax=1;
	int len=0;
	if(dp[i][j]!=0)
		return dp[i][j];
	if(i-1>=1&&high[i][j]>high[i-1][j])
	{
		len=f(i-1,j)+1;
		if(len>lenmax)
			lenmax=len;
	}
	if(i+1<=r&&high[i][j]>high[i+1][j])
	{
		len=f(i+1,j)+1;
		if(len>lenmax)
			lenmax=len;
	}
	if(j-1>=1&&high[i][j]>high[i][j-1])
	{
		len=f(i,j-1)+1;
		if(len>lenmax)
			lenmax=len;
	}
	if(j+1<=c&&high[i][j]>high[i][j+1])
```

```
    {
        len=f(i,j+1)+1;
        if(len>lenmax)
            lenmax=len;
    }
    return lenmax;
}
```

6.鸡蛋的硬度

输入包括多组数据，每组数据一行，包含两个正整数 n 和 m(1<=n<=100,1<=m<=10)，其中 n 表示楼的高度，m 表示你现在拥有的鸡蛋个数，这些鸡蛋硬度相同（即它们从同样高的地方掉下来要么都摔碎要么都不碎），并且小于等于 n。你可以假定硬度为 x 的鸡蛋从高度小于等于 x 的地方摔无论如何都不会碎（没摔碎的鸡蛋可以继续使用），而只要从比 x 高的地方扔必然会碎。

对每组输入数据，你可以假定鸡蛋的硬度在 0 至 n 之间，即在 n+1 层扔鸡蛋一定会碎。

分析： dp[i][j]表示 i 层楼，j 个鸡蛋按照最优策略在最坏情况下所需要的扔鸡蛋次数。如何理解“最优策略在最坏情况下”？ 由于不管从第几层开始，i 层 j 个鸡蛋的情况次数都是相同的，假设在 k 层时，①鸡蛋碎了，那么还要 dp[k-1][j-1]②鸡蛋没碎，那么还要 dp[i-k][j]次，最坏情况就是 max{dp[k-1][j-1]+1，dp[i-k][j]+1},k 从 1 到 i，但是又要最优策略，所以我们就得出 dp[i][j]=min{max{ dp[k-1][j-1]+1，dp[i-k][j]+1}}

```
#include<stdio.h>
int dp[101][11];
int max(int a,int b)
{
    if(a>=b)
        return a;
    else
        return b;
}
int main(void)
{
    int i,j,k;
    int n,m;
    int h,min;
    for(i=1;i<=100;i++)
        dp[i][1]=i;
    for(i=1;i<=10;i++)
        dp[0][i]=0;
    for(i=1;i<=10;i++)
        dp[1][i]=1;
    for(j=2;j<=10;j++)
    {
        for(i=1;i<=100;i++)
        {
            min=10000000;
            for(k=1;k<=i;k++)
            {
                h=max(dp[k-1][j-1]+1,dp[i-k][j]+1);
                if(h<min)
                    min=h;
            }
            dp[i][j]=min;
        }
    }
    while(scanf("%d%d",&n,&m)==2)
        printf("%d\n",dp[n][m]);
    return 0;
}
```

7.放苹果问题

把 M 个同样的苹果放在 N 个同样的盘子里，允许有的盘子空着不放，问共有多少种不同的分法（用 K 表示）？注意：5，1，1 和 1，5，1 是同一种分发。第一行是测试数据的数目 t（0<= t <= 20）,其后的 t 行均包含两个整数 M 和 N，以空格分开。1<= N<= 10　　M＜100
分析：dp[i][j]表示 i 个苹果放到 j 个盘子。分为两种情况：①如果全部盘子都放有苹果，那么，相当于将 i-j 个苹果放到 j 个盘子中，dp[i][j]=dp[i-j][j]，但这里要注意 i=j 时，dp[i][j]=1，i<j 时，dp[i][j]=0；②如果至少有一个空的，那么就相当于把 i 个苹果放到 j-1 个盘子中。综上：当 i>j 时，dp[i][j]=dp[i][j-1]+dp[i-j][j]；当 i=j 时，dp[i][j]=dp[i][j-1]+1；当 i<j 时，dp[i][j]=dp[i][j-1]。

```
#include<stdio.h>
int dp[100][11];
int main(void)
{
	int i,j;
	int t;
	int m,n;
	scanf("%d",&t);
	for(i=1;i<=100;i++)
		dp[i][1]=1;
	for(j=2;j<=10;j++)
	{
		for(i=1;i<=100;i++)
		{
			if(i-j>0)
				dp[i][j]=dp[i][j-1]+dp[i-j][j];
			else if(i-j==0)
				dp[i][j]=dp[i][j-1]+1;
			else
				dp[i][j]=dp[i][j-1];
		}
	}
	while(t--)
	{
		scanf("%d%d",&m,&n);
		printf("%d\n",dp[m][n]);
	}
	return 0;
}
```

8.最大子矩阵
已知矩阵的大小定义为矩阵中所有元素的和。给定一个矩阵，你的任务是找到最大的非空(大小至少是 1 * 1)子矩阵。比如，如下 4 * 4 的矩阵
0 -2 -7 0
9 2 -6 2
-4 1 -4 1
-1 8 0 -2
的最大子矩阵是
9 2
-4 1
-1 8
这个子矩阵的大小是 15。输入是一个 N * N 的矩阵。输入的第一行给出 N (0 < N <= 100)。再后面的若干行中，依次（首先从左到右给出第一行的 N 个整数，再从左到右给出第二行的 N 个整数……）给出矩阵中的 N2 个整数，整数之间由空白字符分隔（空格或者空行）。已知矩阵中整数的范围都在[-127, 127]。
分析：我们将二维最大子矩阵和问题转化为一维最大子序列和问题，从第 i 行(1=<i<=n)开始，到第 j 行（i=<j=<n），这 j-i+1 行中最大子矩阵和都求出来，取最大值。而在求这 j-i+1 行中的最大子矩阵时，我们可以先将每一列加起来，把它转化为一维最大子序列和问题，即可求解。一维最大子序列和问题可以代码如下（时间复杂度 o(n)）。

```
#include<stdio.h>
int n;
```

```
int num[101][101];   //用来储存每个数字
int dp[101][101];    //用来储存到从第 i 行开始第 j 行为止每一列的和
int f(int j)             //求一维数组的最大子序列和
{
    int i,max=0,sum=0;
    for(i=1;i<=n;i++)
    {
        sum+=dp[j][i];
        if(sum>max)
            max=sum;
        else if(sum<0)
            sum=0;
    }
    return max;
}
int main(void)
{
    int i,j,k,x,y,z;
    int max=-100000;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            scanf("%d",&num[i][j]);
    for(i=1;i<=n;i++)                          //从第 i 行开始
    {
        for(j=i;j<=n;j++)                      //到第 j 行
        {
            if(i==j)
            {
                for(k=1;k<=n;k++)
                    dp[j][k]=num[j][k];
                x=f(j);
                if(x>max)
                    max=x;
            }
            else
            {
                for(k=1;k<=n;k++)
                    dp[j][k]=dp[j-1][k]+num[j][k];
                x=f(j);
                if(x>max)
                    max=x;
            }
        }
        for(y=1;y<=n;y++)                  //清零数组
            for(z=1;z<=n;z++)
                dp[y][z]=0;
    }
    printf("%d",max);
    return 0;
}
```

9.采药

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。输入第一行有两个整数 T（1 ≤ T ≤ 1000）和 M（1 ≤ M ≤ 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 1 到 100 之间（包括 1 和 100）的整数，分别表示采摘某株草药的时间和这株草药的价值。

分析：假设我已经找到了总价值最大的采摘方案 dp[M][T]，那么就有两种情况，①如果方案中包括第 M 株药材，则剩余药材总价值必是剩余 M-1 株药材中在 T-t[M]时间内可采得药材的最大值；②如果不包括第 M 株药材，则方案等价于在其余 M-1 株药材中，在时间 T 内可以采得药材最大总价值的方案。dp[i][j]=max(dp[i-1][j-t]+v,dp[i-1][j])

```
#include<stdio.h>
int dp[101][1001]; //M 株药材中在 T 时间的最大价值
int max(int a,int b)
{
    if(a>=b)
        return a;
    else
        return b;
}
int main(void)
{
    int i,j;
    int T,M;        //T 表示总共能用来采药的时间，M 表示草药的总数
    int t,v;        //t 表示采摘每一株的时间，v 表示每一株的价值
    scanf("%d%d",&T,&M);
    for(i=1;i<=M;i++)               //对药材数量的循环
    {
        scanf("%d%d",&t,&v);    //输入每一株的采摘时间和价值
        for(j=1;j<=T;j++)         //对每一种时间的约束
        {
            if(i==1)                //第一株药材
            {
                if(j>=t)          //时间够就摘 ，否则不摘
                    dp[i][j]=v;
                else
                    dp[i][j]=0;
            }
            else                      //如果不是第一株
            {
                if(j>=t)          //时间够
                    dp[i][j]=max(dp[i-1][j-t]+v,dp[i-1][j]);
                else                //时间不够
                    dp[i][j]=dp[i-1][j];
            }
        }
    }
    printf("%d",dp[M][T]);
    return 0;
}
```

10.前缀表达式（逆波兰表达式）

前缀表达式是一种把运算符前置的算术表达式，例如普通的表达式 2 + 3 的前缀表示法为+ 2 3。前缀表达式的优点是运算符之间不必有优先级关系，也不必用括号改变运算次序，例如(2 + 3) * 4 的前缀表示法为* + 2 3 4。本题求解前缀表达式的值，其中运算符包括+ - * /四个。输入为一行，其中运算符和运算数之间都用空格分隔，运算数是浮点数。

分析：利用递归求解。在递归函数中，针对当前的输入有 5 种情况：①输入的是常数，那么表达式的值就是这个常数；②输入的是‘+’，则表达式的值是再继续读入两个表达式并计算出他们的值，然后将他们相加；③输入‘-’④输入‘*’⑤输入‘/’。

```
#include<stdio.h>
#include<stdlib.h>
#include<stdlib.h>
double f()
{
    char a[10];
    scanf("%s",a);
```

```
	switch(a[0])
	{
		case '+':return f()+f();
		case '-':return f()-f();
		case '*':return f()*f();
		case '/':return f()/f();
		default :return atof(a);
	}
}
int main(void)
{
	double ans;
	ans=f();
	printf("%f\n",ans);
	return 0;
}
```

11.大整数乘法（多组）

```
#include<stdio.h>
#include<string.h>
char a[100][500];
char b[100][500];
int c[100][500];
int d[100][500];
int main(void)
{
	int n,m;
	int i,j,k;
	int x;

	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		for(j=0;j<500;j++)
		{
			c[i][j]=0;
			d[i][j]=0;
		}
	}
	for(i=0;i<n;i++)
	{
		scanf("%s",a[i]);
		scanf("%s",b[i]);
	}

	for(i=0;i<n;i++)
	{
		for(j=strlen(a[i])-1,k=0;j>=0;j--)
		{
			c[i][k]=a[i][j]-'0';
			k++;
		}
		for(j=strlen(b[i])-1,k=0;j>=0;j--)
		{
			d[i][k]=b[i][j]-'0';
			k++;
		}
	}
	for(i=0;i<n;i++)
```

```
    {
        for(j=0;j<500;j++)
        {
            if(c[i][j]+d[i][j]>9)
            {
                c[i][j]=c[i][j]+d[i][j]-10;
                c[i][j+1]++;
            }
            else
                c[i][j]+=d[i][j];
        }
    }

    for(i=0,k=0;i<n;i++,x=0)
    {
        for(j=400;j>=0;j--)
        {
            if(x==1)
                printf("%d",c[i][j]);
            else if(c[i][j]>0)
            {
                printf("%d",c[i][j]);
                x=1;
                k=1;
            }
        }
        printf("\n");
    }
    if(x==0&&k==0)
        printf("0");
    return 0;
}
```

12.大整数除法（除以 13 类似）

```
#include<stdio.h>
#include<string.h>
char beichushu[150];
char chushu[150];
int beichu[150];                    //用来存储被除数
int chu[150];                       //用来存储除数（不变的）
int chu1[250];                      //用来存储放大倍数的除数
int shang[200];                     //用来存储商
void subtract(int *,int *);         //大整数减法，将被除数和除数相减
int compare(int *,int *);           //比较两个数的大小
int wei(int *);                     //获得数的位数
void fangda(int *, int);            //将除数放大 10 的 n 次方倍
int main(void)
{
    int i,j,k,n,max;
    scanf("%d",&n);                 //输入要求的组数
    while(n--)
    {
        int x=0;
        int m=0;
        scanf("%s",beichushu);
        scanf("%s",chushu);
        memset(beichu,0,sizeof(beichu));
        memset(chu,0,sizeof(chu));
        memset(chu1,0,sizeof(chu1));
```

```
            memset(shang,0,sizeof(shang));
            for(i=strlen(beichushu)-1,j=0;i>=0;i--)
                beichu[j++]=beichushu[i]-'0';
            for(i=strlen(chushu)-1,j=0;i>=0;i--)
                chu[j++]=chushu[i]-'0';
            for(i=strlen(chushu)-1,j=0;i>=0;i--)
                chu1[j++]=chushu[i]-'0';
            k=wei(beichu)-wei(chu);                  //k 是被除数和除数的位数之差
            max=k;                  //将这个一开始的位数之差赋给 max，用于最后确定商的位数
            int a=compare(beichu,chu);              //判断被除数小于除数，输出 0
            fangda(chu1,k);                          //放大除数
            while(compare(beichu,chu))
            {
                while(compare(beichu,chu1))
                {
                    subtract(beichu,chu1);
                    m++;
                }
                shang[x]=m;
                x++;
                m=0;
                memset(chu1,0,sizeof(chu1));
                for(i=strlen(chushu)-1,j=0;i>=0;i--)
                    chu1[j++]=chushu[i]-'0';
                k--;
                if(k>=1)
                    fangda(chu1,k);
            }
            if(shang[0]==0)
                for(i=1;i<=max;i++)
                    printf("%d",shang[i]);
            else if(shang[0]>0)
                for(i=0;i<=max;i++)
                    printf("%d",shang[i]);
            if(a==0)
                printf("0");
            printf("\n");
        }
        return 0;
}

void subtract(int *beichu,int *chu1)
{
        int i;
        for(i=0;i<120;i++)
        {
            beichu[i]=beichu[i]-chu1[i];
            if(beichu[i]<0)
            {
                beichu[i]+=10;
                beichu[i+1]--;
            }
        }
}
int compare(int *beichu,int *chu1)
{
        if(wei(beichu)>wei(chu1))
            return 1;
```

```
        else if(wei(beichu)<wei(chu1))
                return 0;
        else
        {
                int i;
                for(i=149;;i--)
                {
                        if(beichu[i]>chu1[i])
                                return 1;
                        else if(beichu[i]<chu1[i])
                                return 0;
                }
        }
        return 1;
}
int wei(int *a)
{
        int i,post;
        for(i=149;i>=0;i--)
                if(a[i]>0)
                {
                        post=i+1;
                                return post;
                }
}
void fangda(int *c,int n)
{
        int i;
        for(i=100;i>=0;i--)
                c[i+n]=c[i];
        for(i=0;i<n;i++)
                c[i]=0;
}
```

13.大整数减法

```
#include<stdio.h>
#include<string.h>
char a[200];
char b[200];
int c[200];
int d[200];
int subtract(int *c,int *d);
int main(void)
{
        int i,j,k,x,y;
        int n;
        scanf("%d",&n);
        while(n--)
        {
                scanf("%s",a);
                scanf("%s",b);
                memset(c,0,sizeof(c));
                memset(d,0,sizeof(d));
                x=strlen(a)-1;
                for(i=x,j=0;i>=0;i--)
                {
                        c[j++]=a[i]-'0';
                }
                y=strlen(b)-1;
```

```
            for(i=y,j=0;i>=0;i--)
            {
                d[j++]=b[i]-'0';
            }
            int post=subtract(c,d);
            for(i=post;i>=0;i--)
                printf("%d",c[i]);
            printf("\n");
        }
        return 0;
}
int subtract(int *c,int *d)
{
        int post=0;
        int i=0;
        for(i=0;i<200;i++)
        {
            c[i]-=d[i];
            if(c[i]<0)
            {
                c[i]+=10;
                c[i+1]--;
            }
            if(c[i])
                post=i;          //记录最高位的位置
        }
        return post;
}
```

14.阶乘的精确值

```
//阶乘的精确值，输入不超过 1000 的正整数 n。
#include<stdio.h>
#include<string.h>
int f[3000];
int main(void)
{
        int i,j;
        int n;
        int x,y;
        int post;
        int m;
        int s;
        scanf("%d",&n);
        memset(f,0,sizeof(f));
        f[0]=1;
        for(i=2;i<=n;i++)
        {
            int c=0;
            for(j=0;j<3000;j++)
            {
                s=f[j]*i+c;
                f[j]=s%10;
                c=s/10;
            }
        }
        for(i=2999;;i--)
        {
            if(f[i]!=0)
            {
```

```
                post=i;
                break;
            }
        }
        for(i=post;i>=0;i--)
        {
            printf("%d",f[i]);
        }
        return 0;
}
```

15.归并排序（分治）

输入 n 个数，将这 n 个数从小到大排序。

//归并排序：将数组分成两组 A，B，如果两组组内数据都是有序的，那么就可以很方便的将这两组数据进行排序

//如何让这二组内数据有序？可以将 A，B 再分成两组，以此类推，当分出来的小组只有一个数据的时候，可以认为

//小组内已经达到了有序，然后再合并相邻的两个小组就可以了。这样先递归的分解数列，再合并数列。

```
#include<stdio.h>
int a[1000];
int temp[1000];
//将有二个有序数列 a[first...mid]和 a[mid+1...last]合并。
void merge(int *a,int first,int mid,int last,int *temp)
{
    int i=first,j=mid+1;
    int m=mid,n=last;
    int k=1;
    while(i<=m&&j<=n)
    {
        if(a[i]<=a[j])
            temp[k++]=a[i++];
        else
            temp[k++]=a[j++];
    }
    while(i<=m)
        temp[k++]=a[i++];
    while(j<=n)
        temp[k++]=a[j++];
    for(i=1;i<k;i++)
    {
        a[first+i-1]=temp[i];
    }
}
void divide(int *a,int first,int last,int *temp)
{
    int mid;
    if(first<last)
    {
        mid=(first+last)/2;
        divide(a,first,mid,temp);              //左边有序
        divide(a,mid+1,last,temp);             //右边有序
        merge(a,first,mid,last,temp);       //再将二个有序数列合并
    }
}
int main(void)
{
    int i,j,k;
    int first,mid,last;
```

```
    int n;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        scanf("%d",&a[i]);
    divide(a,1,n,temp);
    for(i=1;i<=n;i++)
        printf("%d ",a[i]);
    return 0;
}
```

16.快速排序

17. 求排列的逆序数（分治归并排序）

输入：第一行是一个整数 n ， 表明该排列有 n 个数（n<=100000) 第二行是 n 个不同的正整数，之间以空格隔开，表明该排列。输出该排列的逆序数

分析：有一种排序的方法是归并排序，归并排序的主要思想是将整个序列分成两部分，分别递归将这两部分排好序之后，再和并为一个有序的序列。在合并的过程中是将两个相邻并且有序的序列合并成一个有序序列，如以下两个有序序列

Seq1：3 4 5 Seq2：2 6 8 9

合并成一个有序序:Seq：2 3 4 5 6 8 9

对于序列 seq1 中的某个数 a[i],序列 seq2 中的某个数 a[j]，如果 a[i]<a[j],没有逆序数，如果 a[i]>a[j]，那么逆序数为 seq1 中 a[i]后边元素的个数(包括 a[i])，即 len1-i+1,

这样累加每次递归过程的逆序数，在完成整个递归过程之后，最后的累加和就是逆序的总数

```
#include<stdio.h>
int a[100001];
int temp[100001];
long long count=0;
//将有二个有序数列 a[first...mid]和 a[mid+1...last]合并。
void merge(int *a,int first,int mid,int last,int *temp)
{
    int i=first,j=mid+1;
    int m=mid,n=last;
    int k=1;
    while(i<=m&&j<=n)
    {
        if(a[i]<=a[j])
            temp[k++]=a[i++];
        else
        {
            temp[k++]=a[j++];
            count+=m-i+1;
        }
    }
    while(i<=m)
    {
        temp[k++]=a[i++];
    }
    while(j<=n)
    {
        temp[k++]=a[j++];
    }
    for(i=1;i<k;i++)
    {
        a[first+i-1]=temp[i];
    }
}
void divide(int *a,int first,int last,int *temp)
{
    int mid;
    if(first<last)
```

```
    {
        mid=(first+last)/2;
        divide(a,first,mid,temp);          //左边有序
        divide(a,mid+1,last,temp);         //右边有序
        merge(a,first,mid,last,temp);      //再将二个有序数列合并
    }
}
int main(void)
{
    int i,j,k;
    int first,mid,last;
    int n;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        scanf("%d",&a[i]);
    divide(a,1,n,temp);
    printf("%lld",count);
    return 0;
}
```

18.多边形游戏

一个多边形，开始有 n 个顶点。每个顶点被赋予一个正整数值，每条边被赋予一个运算符“+”或“*”。所有边依次用整数从 1 到 n 编号。 现在来玩一个游戏，该游戏共有 n 步：

第 1 步，选择一条边，将其删除

随后 n-1 步，每一步都按以下方式操作：（1）选择一条边 E 以及由 E 连接着的 2 个顶点 v1 和 v2； （2）用一个新的顶点取代边 E 以及由 E 连接着的 2 个顶点 v1 和 v2，将顶点 v1 和 v2 的整数值通过边 E 上的运算得到的结果值赋给新顶点。

最后，所有边都被删除，只剩一个顶点，游戏结束。游戏得分就是所剩顶点上的整数值。那么这个整数值最大为多少？

分析：从第一条边开始依次删除，每一次都会有一个 dp[1][n]，取其中最大的 dp[1][n]。删除第 i 条边之后把第 i+1 个数赋值到数组 a 的第一个位置，dp[i][j]=max(dp[i][k]+(*)dp[k+1][j]) k 从 i 到 j。

```
#include<stdio.h>
int dp[100][100];              //dp[1][n]表示删除第 i 条边后所的最大值，最后比较即可
int num[100];                  //存储输入的数字
int a[100];                    //删除第 i 条边后将第 i+1 个数字开始放在数组内
char fh[100];                  //存储输入的运算符
char hf[100];                  //删除第 i 条边后将第 i+1 个运算符开始放在数组内
int main(void)
{
    int i,j,k;
    int n;
    int max=0,max1;
    int m;
    int x,y,z;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        scanf("%d %c",&num[i],&fh[i]);
    for(i=1;i<=n;i++)                          //删除第 i 条边
    {
        k=1;
        for(j=i+1;k<=n;j++)                    //把第 i+1 个数放在新数组第一位
        {
            if(j<=n)
            {
                a[k]=num[j];
                hf[k]=fh[j];
                k++;
            }
```

```
                else                                    //最后一个运算符，我没有删去
                {
                    j=j%n;
                    a[k]=num[j];
                    hf[k]=fh[j];
                    k++;
                }
            }
            for(j=1;j<=n;j++)
                dp[j][j]=a[j];
            for(j=1;j<=n-1;j++)
            {
                for(x=1;x+j<=n;x++)
                {
                    max1=0;
                    for(y=x;y<x+j;y++)
                    {
                        if(hf[y]=='+')
                        {
                            m=dp[x][y]+dp[y+1][x+j];
                            if(m>max1)
                                max1=m;
                        }
                        if(hf[y]=='*')
                        {
                            m=dp[x][y]*dp[y+1][x+j];
                            if(m>max1)
                                max1=m;
                        }
                    }
                    dp[x][x+j]=max1;
                }
            }
            if(dp[1][n]>max)
                max=dp[1][n];
        }
        printf("%d",max);
        return 0;
}
```

19.汉诺塔

```
#include<stdio.h>
void move(int n,char one,char two,char three);
int main(void)
{
    int m;
    scanf("%d",&m);
    move(m,'A','B','C');
    return 0;
}
void move(int n,char one,char two,char three)
{
    if(n==1)
        printf("%c->%c\n",one,three);
    else if(n>1)
    {
        move(n-1,one,three,two);
        printf("%c->%c\n",one,three);
        move(n-1,two,one,three);
```

```
    }
}
```
20.汉诺塔移到中间那根柱子
```
#include<stdio.h>
void move(int n,char one,char two,char three);
int main(void)
{
    int m;
    char a,b,c;
    scanf("%d %c %c %c",&m,&a,&c,&b);
    move(m,a,b,c);
    return 0;

}
void move(int n,char one,char two,char three)
{
    if(n==1)
        printf("%c->%d->%c\n",one,1,three);
    else if(n>1)
    {
        move(n-1,one,three,two);
        printf("%c->%d->%c\n",one,n,three);
        move(n-1,two,one,three);
    }
}
```
21.八皇后问题（输出棋盘）

见计算概论课本；

22.八皇后问题（输出数字）
```
#include<stdio.h>
#include<string.h>
int num[93][9];
int dp[9]={0};
int count=0;
int ok;
void f(int i)
{
    int j,k,l;
    if(i==9)
    {
        count++;
        for(k=1;k<=8;k++)
            num[count][k]=dp[k];
        dp[i-1]=0;
    }
    else
    {
        for(j=1;j<=8;j++)
        {
            ok=1;
            for(k=1;k<i;k++)
            {
                if(dp[k]==j||(i+j)==(k+dp[k])||(i-j)==(k-dp[k]))
                {
                    ok=0;
                    break;
                }
            }
            if(ok==1)
```

```
                {
                    dp[i]=j;
                    f(i+1);
                }
            }
            if(ok==0)                       //撤销上一步
                dp[i-1]=0;
        }
}
int main(void)
{
        int i,j,k,n;
        for(i=1;i<=8;i++)
        {
            dp[1]=i;
            f(2);
        }
        scanf("%d",&n);
        while(n--)
        {
            scanf("%d",&k);
            for(i=1;i<=8;i++)
                printf("%d",num[k][i]);
            printf("\n");
        }
        return 0;
}
```

23.试剂稀释（习题 15-10）

```
#include<stdio.h>
double a[101];
int dp[101];
int main(void)
{
        int i,j,k;
        int n;
        int max;
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
            scanf("%lf",&a[i]);
        }
        dp[n]=0;
        for(i=n-1;i>=1;i--)
        {
            max=0;
            for(j=i+1;j<=n;j++)
            {
                if(a[i]>=a[j]&&dp[j]+1>max)
                    max=dp[j]+1;
            }
            dp[i]=max;
        }
        int m=0;
        for(i=1;i<=n;i++)
        {
            if(dp[i]>m)
                m=dp[i];
        }
```

```
	printf("%d",m+1);
	return 0;
}
```

24.五户共井问题（习题 15-5）

```
#include<stdio.h>
int main(void)
{
	int n1,n2,n3,n4,n5;
	int a,b,c,d,e;
	int k;
	int t;
	int x=0;
	scanf("%d",&k);
	scanf("%d%d%d%d%d",&n1,&n2,&n3,&n4,&n5);
	for(t=1;t<=k*100;t++)
	{

	if((n5*(n4*(n3*(n2-1)+1)-1)+1)*t%(n1*n2*n3*n4*n5+1)==0&&(n5*(n4*(n3*(n2-1)+1)-1)+
1)*t/(n1*n2*n3*n4*n5+1)>0)
		a=(n5*(n4*(n3*(n2-1)+1)-1)+1)*t/(n1*n2*n3*n4*n5+1);
	else
		continue;
	if(t-n1*a>0)
		b=t-n1*a;
	else
		continue;
	if(t-n2*b>0)
		c=t-n2*b;
	else
		continue;
	if(t-n3*c>0)
		d=t-n3*c;
	else
		continue;
	if(t-n4*d>0)
		e=t-n4*d;
	else
		continue;
	printf("%d:%d %d %d %d %d\n",t,a,b,c,d,e);
	x=1;
	}
	if(x==0)
		printf("no answer!");
	return 0;
}
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
	int n1,n2,n3,n4,n5,h;
	int A,B,C,D,E,k;
	int up,down,i,j,x,y,z;
	int a[5],flag=0;
	scanf("%d %d %d %d %d %d",&k,&n1,&n2,&n3,&n4,&n5);
	k=k*100;
	up=n5*(n4*(n3*(n2-1)+1)-1)+1;   //分子
	down=n1*n2*n3*n4*n5+1;              //分母
	x=down;
```

```
        y=up;
        while(y>0)   //求最大公约数约分分子分母
        { z=x%y; x=y; y=z; }
        down=down/x;
        up=up/x;
        if(down>k) printf("not found");
        else
        {
            h=down;
            A=up;
            B=h-n1*A;
            C=n1*n2*A+(1-n2)*h;
            D=0-n1*n2*n3*A+(n3*(n2-1)+1)*h;
            E=n1*n2*n3*n4*A-(n4*(n3*(n2-1)+1)-1)*h;
            a[0]=A;  a[1]=B;  a[2]=C;  a[3]=D;  a[4]=E;
            for(i=0;i<4;i++)for(j=i+1;j<5;j++)
            if(a[i]==a[j]){flag=1;break;}
            if(flag) printf("not found");
            else printf("%d %d %d %d %d %d",h,A,B,C,D,E);
        }
        return 0;
}
```

25.称硬币

赛利有 12 枚银币。其中有 11 枚真币和 1 枚假币。假币看起来和真币没有区别，但是重量不同。但赛利不知道假币比真币轻还是重。于是他向朋友借了一架天平。朋友希望赛利称三次就能找出假币并且确定假币是轻是重。例如:如果赛利用天平称两枚硬币，发现天平平衡，说明两枚都是真的。如果赛利用一枚真币与另一枚银币比较，发现它比真币轻或重，说明它是假币。经过精心安排每次的称量，赛利保证在称三次后确定假币。

```
//依次假设 A~L 是假币并且轻，再依次假设假币并且重，检查是否满足要求。
//如果左盘有这个“假币”，如果右盘有，如果没有。
#include<stdio.h>
#include<string.h>
int main(void)
{
    char total[13];
    char left[3][7];
    char right[3][7];
    char s[3][10];
    int i,j,k;
    int n;
    int count;
    for(i=0;i<12;i++)
        total[i]='A'+i;
    scanf("%d",&n);
    while(n--)
    {
        for(j=0;j<3;j++)
            scanf("%s%s%s",left[j],right[j],s[j]);
        for(i=0;i<12;i++)
        {
            for(j=0;j<3;j++)            //假设轻
            {
                if(strchr(left[j],total[i])!=NULL)        //如果在左盘
                {
                    if(strcmp(s[j],"down")==0)
                        count=1;
                    else
                    {
```

```
                count=0;
                break;
            }
        }
        else if(strchr(right[j],total[i])!=NULL)     //如果在右盘
        {
            if(strcmp(s[j],"up")==0)
                count=1;
            else
            {
                count=0;
                break;
            }
        }
        else
        {
            if(strcmp(s[j],"even")==0)
                count=1;
            else
            {
                count=0;
                break;
            }
        }
    }
    if(count==1)
    {
        printf("%c is the counterfeit coin and it is light.\n",total[i]);
        break;
    }
    for(j=0;j<3;j++)            //假设重
    {
        if(strchr(left[j],total[i])!=NULL)          //如果在左盘
        {
            if(strcmp(s[j],"up")==0)
                count=1;
            else
            {
                count=0;
                break;
            }
        }
        else if(strchr(right[j],total[i])!=NULL)     //如果在右盘
        {
            if(strcmp(s[j],"down")==0)
                count=1;
            else
            {
                count=0;
                break;
            }
        }
        else
        {
            if(strcmp(s[j],"even")==0)
                count=1;
            else
            {
```

```
                        count=0;
                        break;
                    }
                }
            }
            if(count==1)
            {
                printf("%c is the counterfeit coin and it is heavy.\n",total[i]);
                break;
            }
        }
    }
    return 0;
}
```

26.骑士游历

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main(void)
{
    char x1,x2;
    char m;
    int y1,y2;
    int x,y;
    int a,b,c,d;        //a 次（1,2）   b 次（-1,2）   c 次（2,1）   d 次（-2，1）
    int h;
    while(scanf("%c%d %c%d%c",&x1,&y1,&x2,&y2,&m)!=EOF)
    {
        int min=100;
        x=x2-x1;
        y=y2-y1;
        if(x1=='a'&&y1==1&&x2=='b'&&y2==2||x1=='b'&&y1==2&&x2=='a'&&y2==1)
        {
            printf("To get from %c%d to %c%d takes %d knight moves.\n",x1,y1,x2,y2,4);
            continue;
        }
        if(x1=='g'&&y1==2&&x2=='h'&&y2==1||x1=='h'&&y1==1&&x2=='g'&&y2==2)
        {
            printf("To get from %c%d to %c%d takes %d knight moves.\n",x1,y1,x2,y2,4);
            continue;
        }
        if(x1=='b'&&y1==7&&x2=='a'&&y2==8||x1=='a'&&y1==8&&x2=='b'&&y2==7)
        {
            printf("To get from %c%d to %c%d takes %d knight moves.\n",x1,y1,x2,y2,4);
            continue;
        }
        if(x1=='g'&&y1==7&&x2=='h'&&y2==8||x1=='h'&&y1==8&&x2=='g'&&y2==7)
        {
            printf("To get from %c%d to %c%d takes %d knight moves.\n",x1,y1,x2,y2,4);
            continue;
        }
        for(a=-4;a<=4;a++)
        {
            for(b=-4;b<=4;b++)
            {
                for(c=-4;c<=4;c++)
                {
                    for(d=-4;d<=4;d++)
```

```
                        {
                            if(a-b+2*c-2*d==x&&2*a+2*b+c+d==y)
                            {
                                h=abs(a)+abs(b)+abs(c)+abs(d);
                                if(h<min)
                                    min=h;
                            }
                        }
                    }
                }
            }
            printf("To get from %c%d to %c%d takes %d knight moves.\n",x1,y1,x2,y2,min);
        }
        return 0;
}
```

BFS

```
#include<stdio.h>
#include<iostream>
using namespace std;
#include<queue>
int dp[10][10];
int xe,ye;
struct node
{
    int x;
    int y;
    int t;
}start;
int bfs(node s)
{
    queue<node> q;
    node a,b;
    int i;
    int dir[8][2]={{-1,-2},{1,-2},{-1,2},{1,2},{-2,-1},{-2,1},{2,-1},{2,1}};
    q.push(s);
    dp[s.x][s.y]=1;
    while(!q.empty())
    {
        a=q.front();
        q.pop();
        if(a.x==xe&&a.y==ye)
            return a.t;
        for(i=0;i<8;i++)
        {
            int x=a.x+dir[i][0];
            int y=a.y+dir[i][1];
            int t=a.t+1;
            if(x>=1&&x<=8&&y>=1&&y<=8&&dp[x][y]==0)
            {
                b.x=x;
                b.y=y;
                b.t=t;
                q.push(b);
                dp[b.x][b.y]=1;
            }
        }
    }
}
```

```
int main(void)
{
	int i,j,k;
	char x1,x2,c;
	int y1,y2;
	while(scanf("%c%d %c%d%c",&x1,&y1,&x2,&y2,&c)==5)
	{
		for(i=1;i<=8;i++)
			for(j=1;j<=8;j++)
				dp[i][j]=0;
		start.x=x1-'a'+1;
		start.y=y1;
		start.t=0;
		xe=x2-'a'+1;
		ye=y2;
		printf("To get from %c%d to %c%d takes %d knight moves.\n",x1,y1,x2,y2,bfs(start));
	}
	return 0;
}
```

27.母牛繁殖问题

```
//若一头小母牛，从出生起第四个年头开始每年生一头母牛，按此规律，第 n 年时有多少头母牛？
#include<stdio.h>
int f(int n);
int main(void)
{
	int sum;
	int m;
	scanf("%d",&m);
	sum=2*f(m)+f(m-1)+f(m-2);
	printf("%d",sum);
	return 0;
}
int f(int n)
{
	int c;
	if(n==2||n==3)
		c=0;
	else if(n==1)
		c=1;
	else
	{
		c=f(n-1)+f(n-3);
	}
	return c;
}
```

28.特殊日历计算（习题 10-6）

```
#include<stdio.h>
int main(void)
{
	int a[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
	int n;
	long i,j,k;
	long x,y,z;
	int hour,minute,second,day,month,year;
	int hour1,minute1,second1,day1,month1,year1;
	scanf("%d",&n);
	while(n--)
```

```
    {
        x=0;y=0;z=0;
        scanf("%d:%d:%d %d.%d.%d",&hour,&minute,&second,&day,&month,&year);

        x=hour*3600+minute*60+second;
        x=(long)(x*1000/864);
        second1=x%100;
        x/=100;
        minute1=x%100;
        x/=100;
        hour1=x%100;

        for(i=2000;i<year;i++)
        {
            if(i%4==0&&i%100!=0||i%400==0)
                y+=366;
            else
                y+=365;
        }
        if(year%4==0&&year%100!=0||year%400==0)
        {
            for(i=1;i<month;i++)
            {
                y+=a[i];
                if(i==2)
                    y++;
            }
            for(i=1;i<=day;i++)
            {
                y++;
            }
        }
        else
        {
            for(i=1;i<month;i++)
                y+=a[i];
            for(i=1;i<=day;i++)
                y++;
        }
        y=y-1;
        day1=y%100+1;
        y/=100;
        month1=y%10+1;
        y=y/10;
        year1=y;
        printf("%d:%d:%d %d.%d.%d\n",hour1,minute1,second1,day1,month1,year1);
    }
    return 0;
}
```

29.玛雅历（习题 10-7）

```
#include <stdio.h>
#include <string.h>
#define NAMELEN 10
char month1[19][NAMELEN]={"pop","no","zip","zotz","tzec","xul",
                          "yoxkin","mol","chen","yax","zac","ceh","mac",
                          "kankin","muan","pax","koyab","cumhu","uayet"};
char month2[20][NAMELEN]={"imix","ik","akbal","kan","chicchan",
        "cimi","manik","lamat","muluk","ok","chuen","eb","ben",
```

```
            "ix","mem","cib","caban","eznab","canac","ahau"};
int main(void)
{
      char month[NAMELEN];
      int nCases, i, m;
      int day, year, dates;
scanf("%d", &nCases);
      printf("%d\n", nCases);
      for (i = 0; i < nCases; i++)
      {
            scanf("%d. %s %d", &day, month, &year); //读出 Haab  历的年月日
            for(m = 0; m < 19; m++)
                  if (!strcmp(month1[m], month)) break; //找到月份对应的数字
            dates = year*365+m*20+day; //计算距离世界开始的天数，从 0  开始
            printf("%d %s %d\n",1+dates%13,month2[dates%20],dates/260); //输出
      }
      return 0;
}
```

30.不吉利日期（习题 10-3）

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
          int w;
          scanf("%d",&w);
          if(w<=6)w=6-w;
          else w=6;
          if(13%7==w) printf("1\n");
          if(44%7==w) printf("2\n");
          if(72%7==w) printf("3\n");
          if(103%7==w) printf("4\n");
          if(133%7==w) printf("5\n");
          if(164%7==w) printf("6\n");
          if(194%7==w) printf("7\n");
          if(225%7==w) printf("8\n");
          if(256%7==w) printf("9\n");
          if(286%7==w) printf("10\n");
          if(317%7==w) printf("11\n");
          if(347%7==w) printf("12\n");
          return 0;
}
```

31.日历问题（习题 10-1）

给定从公元 2000 年 1 月 1 日(周六)开始逝去的天数，你的任务是给出这一天是哪年哪月哪日星期几

输入：n 输入包含若干行，每行包含一个正整数，表示从 2000 年 1 月 1 日开始逝去的天数。输入最后一行是−1, 不必处理。可以假设结果的年份不会超过 9999。

输出：n 对每个测试样例，输出一行，该行包含对应的日期和星期几。

n 格式为“YYYY-MM-DD DayOfWeek”，其中 “DayOfWeek” 必须是下面中的一个："Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday" and "Saturday“。

样例输入

```
1730
1740
1750
1751
-1
```

样例输出

```
2004-09-26 Sunday
2004-10-06 Wednesday
```

2004-10-16 Saturday
2004-10-17 Sunday

```
#include<stdio.h>
char a[7][10]={"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"};
int y,m,d,n;
int p(int n)
{
	if(n%4==0&&n%100!=0||n%400==0)	return 1;
	else return 0;
}
void fun()
{
	int v;
	for(y=2000,d=0;d<=n;y++)
	{
		if(p(y)) d+=366;
		else d+=365;
	}
	y--;
	if(p(y)) d-=366;
	else d-=365;
	m=1;
	if(p(y)) v=29;
	else v=28;
	while(d<=n)
	{
		switch(m++)
		{
		case 1:
		case 3:
		case 5:
		case 7:
		case 8:
		case 10:
		case 12:d+=31;break;
		case 2:d+=v;break;
		default:d+=30;
		}
	}
	m--;
	switch(m)
	{
		case 1:
		case 3:
		case 5:
		case 7:
		case 8:
		case 10:
		case 12:d-=31;break;
		case 2:d-=v;break;
		default:d-=30;
	}
	d=n-d+1;
	printf("%d-%02d-%02d ",y,m,d);
}
int main()
{
	while(scanf("%d",&n),n!=-1)
```

```
    {
        fun();
        if(m<3)
        {
            y--;
            m+=12;
        }
        printf("%s\n",a[(d+2*m+3*(m+1)/5+y+y/4+y/400-y/100)%7]);
    }
    return 0;
}
```

32.计算两个日期之间的天数（习题 10-2）

```
#include<stdio.h>
int main(void)
{
    int day[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
    int y1,m1,d1,y2,m2,d2;
    int i,j,k;
    int c=0;
    scanf("%d%d%d%d%d%d",&y1,&m1,&d1,&y2,&m2,&d2);
    for(i=y1+1;i<=y2-1;i++)
    {
        if(i%4==0&&i%100!=0||i%400==0)
            c+=366;
        else
            c+=365;
    }
    if(y1==y2)
    {
        while(m1!=m2||d1!=d2)
        {
            c++;
            d1++;
            if(y1%4==0&&y1%100!=0||y1%400==0)
            {
                if(m1==2)
                {
                    if(d1>day[m1]+1)
                    {
                        m1++;
                        d1=1;
                    }
                }
                else
                {
                    if(d1>day[m1])
                    {
                        m1++;
                        d1=1;
                    }
                }
            }
            else
            {
                if(d1>day[m1])
                {
                    m1++;
                    d1=1;
```

```
                }
            }
        }
    }
    else
    {
        while(m1!=13||d1!=1)
        {
            c++;
            d1++;
            if(y1%4==0&&y1%100!=0||y1%400==0)
            {
                if(m1==2)
                {
                    if(d1>day[m1]+1)
                    {
                        m1++;
                        d1=1;
                    }
                }
                else
                {
                    if(d1>day[m1])
                    {
                        m1++;
                        d1=1;
                    }
                }
            }
            else
            {
                if(d1>day[m1])
                {
                    m1++;
                    d1=1;
                }
            }
        }
        i=1;
        j=1;
        while(i!=m2||j!=d2)
        {
            c++;
            j++;
            if(y2%4==0&&y2%100!=0||y2%400==0)
            {
                if(i==2)
                {
                    if(j>day[i]+1)
                    {
                        i++;
                        j=1;
                    }
                }
                else
                {
                    if(j>day[i])
                    {
```

```
                                i++;
                                j=1;
                            }
                        }
                    }
                    else
                    {
                        if(j>day[i])
                        {
                            i++;
                            j=1;
                        }
                    }
                }
            }
        printf("%d",c);
        return 0;
}
```

33.校门外的树（习题 11-2）

```
#include<stdio.h>
#include<stdlib.h>
int main(void)
{
    int L,M;
    int i,j,k;
    int start,end;
    int *p;
    scanf("%d%d",&L,&M);
    p=(int *)malloc((L+1)*sizeof(int));
    for(i=0;i<=L;i++)
        p[i]=1;
    while(M--)
    {
        scanf("%d%d",&start,&end);
        for(i=0;i<=L;i++)
        {
            if(i>=start&&i<=end)
                p[i]=0;
        }
    }
    int count=0;
    for(i=0;i<=L;i++)
    {
        if(p[i]==1)
            count++;
    }
    printf("%d",count);
    return 0;
}
```

34.开关灯

```
#include<stdio.h>
#include<math.h>
int main(void)
{
    int n,i;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        if((int)sqrt(i)*(int)sqrt(i)!=i)
```

```
                printf("%d ",i);
        return 0;
}
```

35.删除数组中的元素（习题 11-4）

```
#include<stdio.h>
int main(void)
{
        int i,j,k=0,n,m;
        scanf("%d",&n);
        int a[n];
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        scanf("%d",&m);
        for(i=0;i<n;)
        {
                if(a[i]==m)
                {
                        for(j=i;j<n;j++)
                                a[j]=a[j+1];
                        k++;
                }
                else
                        i++;
        }
        for(i=0;i<n-k-1;i++)
                printf("%d ",a[i]);
        printf("%d",a[n-k-1]);
        return 0;
}
```

36.最匹配的矩阵（习题 11-7）

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main(void)
{
        int m,n,r,s,max,maxi=0,maxj=0,i,j,k,l,temp=0,**A,**B;
        scanf("%d%d",&m,&n);
        A = (int**)malloc(sizeof(int*)*m);
        for(i=0;i<m;i++){
                A[i]=(int*)malloc(sizeof(int)*n);
                for(j=0;j<n;j++){
                        scanf("%d",&A[i][j]);
                }
        }
        scanf("%d%d",&r,&s);
        B = (int**)malloc(sizeof(int*)*r);
        for(i=0;i<r;i++){
                B[i]=(int*)malloc(sizeof(int)*s);
                for(j=0;j<s;j++){
                        scanf("%d",&B[i][j]);
                }
        }
        max = r*s*100;
        for(k=0;k<=m-r;k++){
                for(l=0;l<=n-s;l++){
                        temp = 0;
                        for(i=k;i<k+r;i++){
                                for(j=l;j<l+s;j++){
```

```
                        temp += fabs(A[i][j]-B[i-k][j-l]);
                    }
                }
                if(temp<max){
                    max=temp;
                    maxi=k;
                    maxj=l;
                }
            }
        }

        for(i=maxi;i<maxi+r;i++){
            for(j=maxj;j<maxj+s;j++){
                printf("%d ",A[i][j]);
            }
            printf("\n");
        }
        return 0;
}
```

37.矩阵归零消减序列和（习题 11-8）

```
#include<stdio.h>
int a[100][100];
int main(void)
{
        int i,j,k,n,m;
        int min;
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
            int sum=0;
            for(j=0;j<n;j++)
                for(k=0;k<n;k++)
                    scanf("%d",&a[j][k]);
            for(m=0;m<n-1;m++)                          //要做 n-1 次
            {
                for(j=0;j<n-m;j++)
                {
                    min=10000;
                    for(k=0;k<n-m;k++)
                        if(a[j][k]<min)
                            min=a[j][k];
                    for(k=0;k<n-m;k++)
                        a[j][k]-=min;
                }
                for(k=0;k<n-m;k++)
                {
                    min=10000;
                    for(j=0;j<n-m;j++)
                        if(a[j][k]<min)
                            min=a[j][k];
                    for(j=0;j<n-m;j++)
                        a[j][k]-=min;
                }
                sum+=a[1][1];
                for(j=2;j<n-m;j++)
                    for(k=0;k<n-m;k++)
                        a[j-1][k]=a[j][k];
                for(k=2;k<n-m;k++)
```

```
                    for(j=0;j<n-m;j++)
                        a[j][k-1]=a[j][k];
            }
            printf("%d\n",sum);
        }
        return 0;
}
```

38.单词替换

```
#include<stdio.h>
#include<string.h>
int main(void)
{
        int i=0,j,k;
        char c;
        char s[200][200];
        char a[200];
        char b[200];
        for(i=0,j=0;;i++,j=0)
        {
            c=getchar();
            while(c!=' '&&c!='\n')
            {
                s[i][j++]=c;
                c=getchar();
            }
            s[i][j]='\0';
            if(c=='\n')
                break;
        }
        scanf("%s",a);
        scanf("%s",b);
        for(j=0;j<=i;j++)
        {
            if(strcmp(s[j],a)==0)
            {
                for(k=0;k<strlen(b);k++)
                    s[j][k]=b[k];
                s[j][k]='\0';
            }
        }
        for(k=0;k<i;k++)
            printf("%s ",s[k]);
        printf("%s",s[k]);
        return 0;
}
```

39.数制转换（2 到 36 进制）

```
#include<stdio.h>
#include<string.h>
long f(int l,int i);
int main(void)
{
        int i,j,k;
        int l,n,r;
        char x[100];
        int y[100];
        long s=0;
        char z[100];
```

```
    scanf("%d%s%d",&l,x,&r);
    memset(y,0,sizeof(y));
    for(i=strlen(x)-1,j=0;i>=0;i--)
    {
        if(x[i]>='0'&&x[i]<='9')
            y[j++]=x[i]-'0';
        else if(x[i]>='A'&&x[i]<='Z')
            y[j++]=x[i]-'A'+10;
        else
            y[j++]=x[i]-'a'+10;
    }
    for(i=0;i<strlen(x);i++)
    {
        s+=y[i]*f(l,i);
    }
    for(i=0;s>=r;i++)
    {
        n=s%r;
        if(n>=0&&n<=9)
            z[i]=n+'0';
        else if(n>=10&&n<=35)
            z[i]=n-10+'A';
        s=s/r;
    }
    if(s!=0&&s>=0&&s<=9)
        z[i++]=s+'0';
    else if(s!=0&&s>=10&&s<=35)
        z[i++]=s-10+'A';
    z[i]='\0';
    if(x[0]-'0'==0)
        printf("%d",0);
    else
    for(i=strlen(z)-1;i>=0;i--)
        printf("%c",z[i]);
    return 0;

}
long f(int l,int i)
{
    int k;
    long count=1;
    for(k=0;k<i;k++)
    {
        count*=l;
    }
    return count;
}
```

40.行程长度编码（习题 12-5）

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char in[1000];
    int i,num=1,a,b;
    scanf("%s",in);
    for(i=0;in[i]!='\0';i++)
    {
        a=in[i];
```

```
            b=in[i+1];
            if(a==b||a-b==32||a-b==-32) num++;
            else
            {
                if(in[i]<91) printf("(%c,%d)",in[i],num);
                else printf("(%c,%d)",in[i]-32,num);
                num=1;
            }
        }
        return 0;
}
```

41.回文子串（习题 12-6）

```
#include<stdio.h>
#include<string.h>
int a[600];         //a[i]存储第 i 个回文子串的起始位置
int b[600];         //b[i]存储第 i 个回文子串的结束位置
char s[600];
int main(void)
{
    int i,j,k,m,n;
    int x,y;
    fgets(s,600,stdin);
    m=strlen(s)-2;
    for(i=0,x=0,y=0;i<=m;i++)
    {
        for(j=0;i-j>=0&&i+j<=m;j++)       //长度为奇数情况
        {
            if(s[i-j]!=s[i+j])
                break;
            else
            {
                a[x]=i-j;
                b[x]=i+j;
                x++;
            }
        }
        for(j=0;i-j>=0&&i+j+1<=m;j++)         //长度为偶数情况
        {
            if(s[i-j]!=s[i+j+1])
                break;
            else
            {
                a[x]=i-j;
                b[x]=i+j+1;
                x++;
            }
        }
    }
    for(k=2;k<=500;k++)
    {
        for(i=0;i<=x;i++)
        {
            if(b[i]-a[i]+1==k)
            {
                for(n=a[i];n<=b[i];n++)
                    printf("%c",s[n]);
                printf("\n");
            }
```

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

42.478-3279（习题 12-9）

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define UNCODE(a) (a>='A'?((a>'Q'?(a-'A'-1):(a-'A'))/3+2):a-'0')
int uncode(char* s)
{
        int i;
        int result=0;
        for(i=0;i<strlen(s);i++)
        {
                if(s[i]=='-') continue;
                result=result*10+UNCODE(s[i]);
        }
        return result;
}
int cmp(const void* a,const void* b)
{
        return *(int*)a-*(int*)b;
}
int main(void)
{
        int* store;
        char s[50];
        int time;
        int n=0;
        int i;
        int start,value;
        int dup=0;
        scanf("%d",&time);
        store=(int*)malloc(sizeof(int)*(time));
        for(n=0;n<time;n++)
        {
                scanf("%s",s);
                store[n]=uncode(s);
        }
        qsort(store,time,sizeof(store[0]),cmp);
        value=store[0];
        start=0;
        for(n=0;n<time;n++)
        {
                if(store[n]!=value){
                        if((n-start)>1){
                                printf("%03d-%04d %d\n",store[start]/10000,store[start]%10000,n-start);
                                dup=1;
                        }
                        start=n;
                        value=store[n];
                }
        }
        if((n-start)>1){
                                printf("%03d-%04d %d\n",store[start]/10000,store[start]%10000,n-start);
                                dup=1;
                        }
```

```
        if(!dup)
        {
                printf("No duplicates.\n");
        }
        free(store);
        return 0;
}
```

43.选择你喜爱的水果（12-10）

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
char fruit[100][100]={"apples","bananas","peaches","cherries","pears","oranges","strawberries"};
char b[20]={"Brussels sprouts"};
char s[1000][1000];
int flag;
int p=0;
void f(int i)
{
        char x[1000][1000];
        char y[1000][1000];
        int j,k;
        int h;
        for(j=0;j<strlen(s[i]);j++)
        {
                x[p][j]=s[i][j];
        }
        h=j-1;
        x[p][j]='\0';
        for(j=0;j<strlen(s[i]);j++)
        {
                if(s[i][j]>='a'&&s[i][j]<='z'||s[i][j]>='A'&&s[i][j]<='Z')
                {
                        y[p][j]=s[i][j];
                }
        }
        y[p][j]='\0';
        for(k=0;k<7;k++)
        {
                if(strcmp(y[p],fruit[k])==0)
                {
                        for(j=0;j<16;j++)
                                s[i][j]=b[j];
                        if(x[p][h]<'a'||x[p][h]>'z')
                        {
                                s[i][j]=x[p][h];
                                j++;
                        }
                        s[i][j]='\0';
                        flag=1;
                        break;
                }
        }
        p++;
}
int main(void)
{
        int i,j,k=0;
        int m;
```

```
	char c;
	while(scanf("%s",s[k++])!=EOF)
	{
		c=getchar();
		if(c=='\n')
		{
			flag=0;
			for(i=0;i<k;i++)
				f(i);
			if(flag==1)
			{
				for(i=0;i<k-1;i++)
				{
					printf("%s ",s[i]);
				}
				printf("%s\n",s[i]);
			}
			else
				printf("You must not enjoy fruit.\n");
			k=0;
		}
	}
	return 0;
}
```

45.n-gram 串频统计（习题 12-7）

```
#include<stdio.h>
#include<string.h>
char a[500];
int post[500];
int main(void)
{
	int i,j,k;
	int n;
	int x=1,y;
	int max=0;
	int count=0;
	int h=0;
	scanf("%d",&n);
	scanf("%s",a);
	for(i=0;i<=strlen(a)-n;i++,count=0)
	{
		for(j=i;j<=strlen(a)-n;j++,x=1)
		{
			for(k=0;k<n;k++)
			{
				if(a[i+k]!=a[j+k])
				{
					x=0;
					break;
				}
			}
			if(x==1)
			{
				count++;
			}
		}
		post[i]=count;
		if(count>=max)
```

```
            {
                max=count;
            }
        }
        if(max<=1)
        {
            printf("NO\n");
        }
        else
        {
            printf("%d\n",max);
            for(j=0;j<i;j++)
            {
                if(post[j]==max)
                {
                    for(k=j;k<j+n-1;k++)
                    {
                        printf("%c",a[k]);
                    }
                    printf("%c\n",a[k]);
                }
            }
        }
        return 0;

}
```

46.电池的寿命（习题 13-3）

```
#include<stdio.h>
#include<stdlib.h>
int comp(const void *c,const void *d)
{
    return *(int *)d-*(int *)c;
}
int a[2000];
int main(void)
{
    int i,j=0,k;
    int n;
    double sum;
    while(scanf("%d",&n)!=EOF)
    {
        sum=0.0;
        for(i=0;i<n;i++)
            scanf("%d",&a[i]);
        qsort(a,n,sizeof(a[0]),comp);
        for(i=1;i<n;i++)
            sum+=(double)a[i];
        if(sum>=a[0])
            printf("%.1lf\n",(double)(sum+a[0])/2);
        else
            printf("%.1lf\n",sum);
    }
    return 0;
}
```

47.找第一个只出现一次的字符（习题 13-8）

```
#include<stdio.h>
#include<string.h>
char s[1000][100000];
```

```
int main(void)
{
	int j,k;
	char i;
	int t;
	int count;
	int min,min1;
	int flag;
	scanf("%d",&t);
	while(t--)
	{
		flag=0;
		min=100000;
		scanf("%s",s[t]);
		for(i='a';i<='z';i++)
		{
			count=0;
			min1=100000;
			for(j=0;j<strlen(s[t]);j++)
			{
				if(i==s[t][j])
				{
					count++;
					if(j<min1)
						min1=j;
				}
			}
			if(count==1)
			{
				if(min1<min)
					min=min1;
				flag=1;
			}
		}
		if(flag==0)
			printf("no\n");
		else
			printf("%c\n",s[t][min]);
	}
	return 0;
}
```

48.第二个重复出现的数（习题 13-10）

```
#include<stdio.h>
int num[1000][600];
int main(void)
{
	int i,j,k;
	int n;
	int m;
	int h=0;
	scanf("%d",&n);
	for(i=1;i<=n;i++)
	{
		h=0;
		scanf("%d",&m);
		for(j=1;j<=m;j++)
		{
			scanf("%d",&num[i][j]);
```

```
            }
            int flag=0;
            int y=0;
            for(j=1;j<=m;j++)
            {
                for(k=1;k<=m;k++)
                {
                    if(num[i][j]==num[i][k]&&flag==1&&num[i][j]!=y&&j!=k)
                    {
                        printf("%d\n",num[i][j]);
                        h=1;
                        break;
                    }
                    if(num[i][j]==num[i][k]&&flag==0&&j!=k)
                    {
                        y=num[i][j];
                        flag=1;
                    }
                }
                if(h==1)
                    break;
            }
            if(h==0)
                printf("NOT EXIST\n");
        }
        return 0;
}
```

49.挑选序列（习题 13-11）

```
#include<stdio.h>
#include<string.h>
int num[1000][1000];
int main(void)
{
        int i=0,j=0,k;
        char c;
        int m;
        int max=0;
        int x,y;
        for(i=0;i<1000;i++)
            for(j=0;j<1000;j++)
                num[i][j]=-1;
        i=0;
        j=0;
        while(scanf("%d",&num[i][j++])!=EOF)
        {
            c=getchar();
            if(c=='\n')
            {
                for(k=0;k<j;k++)
                {
                    if(num[i][k]>=max)
                    {
                        max=num[i][k];
                        x=i;
                        y=k;
                    }
                }
                i++;
```

```
                j=0;
            }
        }
        for(m=0;num[x][m]!=-1;m++)
        ;
        for(k=0;k<m-1;k++)
            printf("%d ",num[x][k]);
        printf("%d",num[x][k]);
        return 0;
}
```

50.白细胞计数（习题 13-12）

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
double a[400];
int main(void)
{
        int i,j,k;
        int n;
        int x,y;
        double l=0;
        double sum=0;
        double average;
        double max=0;
        double min=100000.0;
        scanf("%d",&n);
        for(i=0;i<n;i++)
            scanf("%lf",&a[i]);
        for(i=0;i<n;i++)
        {
            if(a[i]>max)
            {
                max=a[i];
                x=i;
            }
            if(a[i]<min)
            {
                min=a[i];
                y=i;
            }
        }
        for(i=0;i<n;i++)
            sum+=a[i];
        sum=sum-min-max;
        average=sum/(n-2);
        for(i=0;i<n;i++)
        {
            if(i==x||i==y)
                ;
            else
            {
                if(fabs(a[i]-average)>l)
                    l=fabs(a[i]-average);
            }
        }
        printf("%.2lf %.2lf",average,l);
        return 0;
}
```

51.受限时间内的最小通行费（习题 13-13）

```
#include<stdio.h>
int a[1000][1000];
int dp[1000][1000];
int min(int c,int d)
{
	if(c>d)
		return d;
	else
		return c;
}
int main(void)
{
	int i,j,k;
	int n;
	scanf("%d",&n);
	for(i=1;i<=n;i++)
		for(j=1;j<=n;j++)
			scanf("%d",&a[i][j]);
	dp[n][n]=a[n][n];
	for(k=2*n-1;k>=2;k--)
	{
		for(i=n;i>=1;i--)
		{
			if(i==n)
				dp[i][k-i]=dp[i][k-i+1]+a[i][k-i];
			else if(k-i==n)
				dp[i][k-i]=dp[i+1][k-i]+a[i][k-i];
			else if(k-i>=1&&k-i<=n)
				dp[i][k-i]=min(dp[i][k-i+1]+a[i][k-i],dp[i+1][k-i]+a[i][k-i]);
		}
	}
	printf("%d",dp[1][1]);
	return 0;
}
```

52.字符排序（习题 14-2）

```
#include<stdio.h>
#include<string.h>
int main(void)
{
	char a[1000];
	char b[1000];
	int i,j,k;
	gets(a);
	char h;
	k=strlen(a);
	if(k%2==1)
	{
		for(i=0;i<=(k-1)/2-1;i++)
		{
			for(j=i+1;j<=(k-1)/2-1;j++)
			{
				if(a[i]<a[j])
				{
					h=a[i];
					a[i]=a[j];
					a[j]=h;
				}
```

```
                }
            }
            for(i=0;i<=(k-1)/2-1;i++)
            {
                b[i]=a[i];
                a[i]=a[i+(k+1)/2];
                a[i+(k+1)/2]=b[i];
            }
            puts(a);
            return 0;
        }
        else
        {
            for(i=0;i<=k/2-1;i++)
            {
                for(j=i+1;j<=k/2-1;j++)
                {
                    if(a[i]<a[j])
                    {
                        h=a[i];
                        a[i]=a[j];
                        a[j]=h;
                    }
                }
            }
            for(i=0;i<=k/2-1;i++)
            {
                b[i]=a[i];
                a[i]=a[i+k/2];
                a[i+k/2]=b[i];
            }
            puts(a);
            return 0;
        }
}
```

53.距离排序（习题 14-4）

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
typedef struct Point
{
    int x;
    int y;
    int z;
}Point;   //定义一个坐标结构体
double Distance(Point *p1,Point *p2)
{
    int x=(p1->x)-(p2->x);
    int y=(p1->y)-(p2->y);
    int z=(p1->z)-(p2->z);
    double temp=(double)(x*x+y*y+z*z);
    return sqrt(temp);
}//求两点间的距离
void Input(Point *p[],int n)
{
    int i;
    for(i=0;i<n;i++)
    {
```

```
            p[i]=(Point *)malloc(sizeof(Point));
            scanf("%d",&p[i]->x);
            scanf("%d",&p[i]->y);
            scanf("%d",&p[i]->z);
        }
}//输入 n 个点的坐标
void Output(Point *p1,Point *p2)
{
    printf("(%d,%d,%d)-(%d,%d,%d)=%.2f\n",p1->x,p1->y,p1->z,p2->x,p2->y,p2->z,Distance(p
1,p2));
}
int main(void)
{
    int n,i,j;
    scanf("%d",&n);
    Point *p[n];
    Input(p,n);
    double a[n*(n-1)/2];
    int k=0;
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            int t=0;
            while(t<k+1 &&(a[t]-Distance(p[i],p[j])>0.00001
||Distance(p[i],p[j])-a[t]>0.00001))t++;
            if(t==k+1){    a[k]=Distance(p[i],p[j]);   k++; }
            //Output(p[i],p[j]);
        }
    }//把所有可能的距离值存放在数组中（不重复）
    int count=k;//数组元素个数
    for(i=0;i<count;i++)
    {
        for(j=i+1;j<count;j++)
        {
            if(a[i]<a[j])
            {    double temp=a[i]; a[i]=a[j]; a[j]=temp;  }
        }
    }//数组 a[]排序
    k=0;
    while(k<count)
    {
        for(i=0;i<n;i++)
        {
            for(j=i+1;j<n;j++)
            {
                if(a[k]>Distance(p[i],p[j]))
                {
                    if(a[k]-Distance(p[i],p[j])<0.000001)//a[k]==Distance(p[i],p[j])
                    Output(p[i],p[j]);
                }
                else
                {
                if(Distance(p[i],p[j])-a[k]<0.000001)//a[k]==Distance(p[i],p[j])
                    Output(p[i],p[j]);
                }//float 型数比较大小比较麻烦，这里是相等的比较
            }
        }
```

```
            k++;
        }//按距离由大到小依次输出两个点的坐标及它们之间的距离
        return 0;
}
```

54.输出前 k 大的数（习题 14-5）

```
#include<stdio.h>
#include<stdlib.h>
int comp(const void *c,const void *d)
{
        return *(int *)d-*(int *)c;
}
int main(void)
{
        int a[1000];
        int i,j,k;
        int n;
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        scanf("%d",&k);
        qsort(a,n,sizeof(a[0]),comp);
        for(i=0;i<k;i++)
                printf("%d\n",a[i]);
        return 0;
}
```

55.区间合并（习题 14-8）

```
#include <stdio.h>
int main(void)
{
        int q[10000];
        int m[10000];
        int i,h,a,b=0,p,s=0;
        double l;
        scanf("%d",&h);
        for (i=0;i<h;i++){
        scanf("%d%d",&q[i],&m[i]);
        }
        a=q[0];
        for(i=0;i<h;i++){
                if (q[i]<a){
                        a=q[i];
                }
                }
        for(i=0;i<h;i++){
                if (m[i]>b){
                        b=m[i];
                }
                }
        for(l=a+0.5;l<b;l++){
                p=0;
                for(i=0;i<h;i++){
                        if((l>=q[i])&&(l<=m[i])){
                                p++;
                        }
                }
                if(p>0)
```

```
            s++;
        }
        if(s==b-a){
            printf("%d %d\n",a,b);
        }else{
        printf("no");
        }
        return 0;
}
```

56.找和为 k 的两个元素（习题 14-9）

```
#include<stdio.h>
int main(void)
{
        int i,j,k;
        int n;
        int a[1001];
        scanf("%d%d",&n,&k);
        for(i=1;i<=n;i++)
            scanf("%d",&a[i]);
        for(i=1;i<=n;i++)
        {
            for(j=i+1;j<=n;j++)
            {
                if(a[i]+a[j]==k)
                {
                    printf("yes");
                    return 0;
                }
            }
        }
        printf("no");
        return 0;
}
#include <stdlib.h>
#include <stdio.h>
int main()
{
        int x1,y1,x2,y2;
        int i,j,k;
        int a[100][100]={0},b[100][100]={0},c[100][100]={0};
        scanf("%d %d",&x1,&y1);
        for(i=0;i<x1;i++)
            for(j=0;j<y1;j++) scanf("%d",&a[i][j]);
        scanf("%d %d",&x2,&y2);
        for(i=0;i<x2;i++)
            for(j=0;j<y2;j++) scanf("%d",&b[i][j]);
        for(i=0;i<x1;i++)
            for(j=0;j<y2;j++)
                for(k=0;k<x2;k++) c[i][j]=c[i][j]+a[i][k]*b[k][j];
        for(i=0;i<x1;i++)
        {
            for(j=0;j<y2-1;j++) printf("%d ",c[i][j]);
            printf("%d\n",c[i][j]);
        }
        return 0;
}
```

58.二维数组回形遍历（习题 6-3）

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

```
#include<string.h>
int num[101][101];
int a[101][101];
int main(void)
{
	int i,j,k,x,y,row,col,n;
	scanf("%d%d",&row,&col);
	for(i=1;i<=row;i++)
		for(j=1;j<=col;j++)
			scanf("%d",&num[i][j]);
	memset(a,0,sizeof(a));
	n=1;
	i=1;
	j=1;
	a[1][1]=1;
	printf("%d\n",num[1][1]);
	while(n<=row*col)
	{
		while(j<col&&a[i][j+1]==0)
		{
			printf("%d\n",num[i][++j]);
			a[i][j]=1;
			n++;
		}
		while(i<row&&a[i+1][j]==0)
		{
			printf("%d\n",num[++i][j]);
			a[i][j]=1;
			n++;
		}
		while(j>1&&a[i][j-1]==0)
		{
			printf("%d\n",num[i][--j]);
			a[i][j]=1;
			n++;
		}
		while(i>1&&a[i-1][j]==0)
		{
			printf("%d\n",num[--i][j]);
			a[i][j]=1;
			n++;
		}
	}
	return 0;
}
```

59.二维数组右上左下遍历（习题 6-4）

```
#include<stdio.h>
int num[101][101];
int main(void)
{
	int i=1,j=2,k=2,x,y,row,col,n=1;
	scanf("%d%d",&row,&col);
	for(x=1;x<=row;x++)
		for(y=1;y<=col;y++)
			scanf("%d",&num[x][y]);
	printf("%d\n",num[1][1]);
	while(n<row*col)
	{
```

```
            int flag=0;
            while(j>=1&&i<=row&&j<=col)
            {
                printf("%d\n",num[i][j]);
                i++;
                j--;
                n++;
                flag=1;
            }
            if(flag==1)
            k++;
            if(k<=col)
            {
                i=1;
                j=k;
            }
            else
            {
                i=k-col+1;
                j=col;
            }
        }
        return 0;
}
```

60.矩阵交换行（习题 6-5）

```
#include<stdio.h>
int end(int (* str)[5],int n,int m)
{
        int t;
        if(n<5&&m<5&&n>=0&&m>=0)
        {
            for(int i=0;i<5;i++)
            {
                t=*(*(str+n)+i);
                *(*(str+n)+i)=*(*(str+m)+i);
                *(*(str+m)+i)=t;
            }
            return 1;
        }
        else
            return 0;
}
int main(void)
{
        int str[5][5];
        int n,m,i,j,f;
        for(i=0;i<5;i++)
            for(j=0;j<5;j++)
                scanf("%d",&str[i][j]);
        scanf("%d %d",&n,&m);
        f=end(str,n,m);
        if(f==0)
            printf("error\n");
        else
        {
            for(i=0;i<5;i++)
            {
                for(j=0;j<4;j++)
```

```
                printf("%d ",str[i][j]);
            printf("%d\n",str[i][j]);
        }
    }
    return 0;
}
```

61.哥德巴赫猜想

```
#include<stdio.h>
int isprime(int i);
int main(void)
{
    int i,j,k;
    int n;
    scanf("%d",&n);
    for(i=6;i<=n;i+=2)
    {
        for(j=2;j<=i/2;j++)
        {
            if(isprime(j)&&isprime(i-j))     //看看 i 分解的所有部分是不是都是素数
            {
                printf("%d=%d+%d\n",i,j,i-j);
                break;
            }
        }
    }
    return 0;
}
int isprime(int i)
{
    int j;
    if(i%2==0)
        return 0;
    for(j=3;j*j<=i;j+=2)
    {
        if(i%j==0)
            return 0;
    }
    return 1;
}
```

62.输出前 n 个素数（优化）

```
//素数枚举方法可以进一步优化，只要验证奇数 i 不能被【2，sqrt（i）】之间的所有素数整除即可
#include<stdio.h>
#include<malloc.h>
int main(void)
{
    int n,i,j;
    //int *primes;
    int k=0;
    scanf("%d",&n);

    //primes=(int *)malloc(sizeof(int)*n);
    int primes[n];
    primes[k++]=2;
    printf("2\n");
    for(i=3;k<n;i+=2)
    {
        for(j=0;primes[j]*primes[j]<=i;j++)
```

```
            {
                if(i%primes[j]==0)
                {
                    break;
                }
            }
            if(primes[j]*primes[j]>i)
            {
                printf("%d\n",i);
                primes[k++]=i;
            }
        }
        //free(primes);
        return 0;
}
```

63.社会名流

```
#include<stdio.h>
int a[2][10000];
int main(void)
{
        int n;
        int i,j,k;
        int x=0,y=0;
        int ok=0;
        scanf("%d",&n);
        scanf("%d%d",&i,&j);
        while(i!=0||j!=0)
        {
            a[0][i]++;
            a[1][j]++;
            scanf("%d%d",&i,&j);
        }
        for(i=0;i<n;i++)
        {
            if(a[0][i]==0&&a[1][i]==n-1)            //一列存储左边一列    一列存储右边一列
            {
                printf("%d\n",i);
                ok=1;
                break;
            }
        }
        if(ok==0)
            printf("NOT FOUND");
        return 0;
}
```

64.蛇形填充数组

```
#include<stdio.h>
int num[1000][1000];
int main(void)
{
        int n,i=1,j=2,k,m=2;
        scanf("%d",&n);
        num[1][1]=1;
        while(m<=n*n)
        {
            while(i<=n&&j>=1)
            {
                num[i][j]=m;
```

```
            m++;
            i++;
            j--;
        }
        if(i>n)
        {
            i=n;
            j=j+2;
        }
        else
            j=1;
        while(i>=1&&j<=n)
        {
            num[i][j]=m;
            m++;
            i--;
            j++;
        }
        if(j>n)
        {
            i=i+2;
            j=j-1;
        }
        else
            i=1;
    }
    for(i=1;i<=n;i++)
    {
        for(j=1;j<n;j++)
            printf("%d ",num[i][j]);
        printf("%d\n",num[i][j]);
    }
    return 0;
}
```

65.最大质因子序列（习题 9-1）

```
#include<stdio.h>
int f(int i);
int main(void)
{
    int i,m,n;
    scanf("%d%d",&m,&n);
    for(i=m;i<n;i++)
        printf("%d,",f(i));
    printf("%d",f(i));
    return 0;
}
int f(int i)
{
    int j,k;
    int max=0;
    if(i==2)
        return 2;
    else if(i==3)
        return 3;
    else
    {
        k=i;
        for(j=2;j<k;j++)
```

```
            {
                while(i%j==0)
                {
                    i/=j;
                    max=j;
                }
            }
        }
        if(max==0)
            max=i;
        return max;
}
```

66.整数的质因子（习题 9-2）

```
#include<stdio.h>
int isprime(int i);
int main(void)
{
        int i,j,k;
        scanf("%d",&k);
        for(i=2;i<=k;i++)
            if(isprime(i)&&k%i==0)
                printf("%d\n",i);
        return 0;
}
int isprime(int i)
{
        int j;
        for(j=2;j*j<=i;j++)
        {
            if(i%j==0)
                return 0;
        }
        return 1;
}
```

67.因子分解（习题 9-3）

```
#include<stdio.h>
int isprime(int i)
{
        int j;
        for(j=2;j*j<=i;j++)
        {
            if(i%j==0)
                return 0;
        }
        return 1;
}
int main(void)
{
        int i,j=0,k;
        int n;
        int a[101],b[101]={0};
        scanf("%d",&n);
        int count;
        for(i=2;i<=n;i++)
        {
            if(isprime(i))
            {
                while(n%i==0)
```

```
                {
                    a[j]=i;
                    n/=i;
                    b[j]++;
                }
                if(b[j]!=0)
                    j++;
            }
        }
        for(k=0;k<j-1;k++)
        {
            if(b[k]>1)
                printf("%d^%d*",a[k],b[k]);
            else if(b[k]==1)
                printf("%d*",a[k]);
        }
        if(b[k]>1)
            printf("%d^%d",a[k],b[k]);
        else if(b[k]==1)
            printf("%d",a[k]);
        return 0;
}
```

68.区间内的真素数（习题 9-4）

```
#include<stdio.h>
int num[100];
int a[10000];
int isprime(int i)
{
        int j;
        for(j=2;j*j<=i;j++)
        {
            if(i%j==0)
                return 0;
        }
        return 1;
}
int f(int i)
{
        int j=1,k=0;
        while(i>=10)
        {
            num[j++]=i%10;
            i/=10;
        }
        num[j]=i;
        for(i=1;i<=j;i++)
        {
            k=k*10+num[i];
        }
        return k;
}
int main(void)
{
        int m,n;
        int i,j=1,flag=0;
        scanf("%d%d",&m,&n);
        for(i=m;i<=n;i++)
        {
```

```
            if(isprime(i)&&isprime(f(i)))
            {
                a[j++]=i;
                flag=1;
            }
        }
        if(flag==0)
            printf("no");
        else
        {
            for(i=1;i<j-1;i++)
                printf("%d,",a[i]);
            printf("%d",a[i]);
        }
        return 0;
}
```

69.分解整数为质因数连乘

```
#include<stdio.h>
int isprime(int i);
int main(void)
{
        int i,j,k;
        int m,n;
        int ok;

        scanf("%d",&n);
        while(n--)
        {
            j=0;
            ok=0;
            scanf("%d",&k);
            printf("%d=",k);
            if(k%2==0)
            {
                printf("2");
                ok=1;
            }
            while(k%2==0)
            {
                j++;
                k=k/2;
            }
            for(i=1;i<j;i++)
                printf("*2");
            for(i=3;i<=k;i+=2)
            {
                j=0;
                if(isprime(i))
                {
                    while(k%i==0)
                    {
                        j++;
                        k=k/i;
                    }
                    if(ok==0&&j>0)
                    {
                        printf("%d",i);
                        for(m=2;m<=j;m++)
```

```
                        printf("*%d",i);
                        ok=1;
                    }
                    else
                        for(m=1;m<=j;m++)
                        printf("*%d",i);
                }
                else
                    continue;
            }
            printf("\n");
        }
        return 0;
}
int isprime(int i)
{
        int j;
        for(j=2;j*j<=i;j++)
        {
            if(i%j==0)
                return 0;
        }
        return 1;
}
```

70.最小公倍数（习题 9-8）

```
#include<stdio.h>
#include<string.h>
int gcd(int ,int );                //辗转相除法求最大公约数
int main(void)
{
        int i,j,k;
        int a,b;
        int max;
        int min;
        scanf("%d,%d",&a,&b);
        if(a>b)
        {
            max=a;
            min=b;
        }
        else if(a<b)
        {
            max=b;
            min=a;
        }
        else
        {
            printf("%d",a);
            return 0;
        }
        printf("%d",a*b/gcd(max,min));
        return 0;
}
int gcd(int max,int min)
{
        int r=1;
        while(r!=0)
        {
```

```
            r=max%min;
            max=min;
            if(r==0)
                  break;
            min=r;
      }
      return min;
}
```

71.约瑟夫问题

```
#include<stdio.h>
int f(int n,int m);
int main(void)
{
      int i,j,k;
      int m,n;
      scanf("%d%d",&n,&m);
      while(n!=0||m!=0)
      {
            printf("%d\n",f(n,m)+1);
            scanf("%d%d",&n,&m);
      }
      return 0;
}
int f(int n,int m)
{
      int k=0;
      int i,j;
      for(i=2;i<=n;i++)              //假设在 n-1 的约瑟夫问题中，最后的获胜者的下标为 k，那么在 n 中，下标应为 k=(m%n+k)%n
      {
            k=(m%i+k)%i;
      }
      return k;
}
```

72.田忌赛马

```
//先比较田忌最慢的马和齐王最慢的马:
//如果田忌最慢的马比齐王最慢的马快，那就比；
//如果田忌最慢的马比齐王最慢的马慢，那田忌的最慢马和齐王的最快马比
//如果田忌最慢的马和齐王最慢的马一样快，那么比较最快的马：
//如果田忌最快的马比齐王最快的马快，那么比；
//如果田忌最快的马比齐王最快的马慢，那么就让田忌最慢马和齐王最快马比；
//如果田忌最快的马和齐王最快的马快，那么就让田忌最慢马和齐王最快马比；
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int tian[1001];
int qi[1001];
int comp(const void *a,const void *b);
int main(void)
{
      int i,j,k;
      int m,n;
      while(scanf("%d",&n)==1&&n!=0)
      {
            int count=0;
            for(i=0;i<n;i++)
                  scanf("%d",&tian[i]);
            for(i=0;i<n;i++)
```

```
            scanf("%d",&qi[i]);
        qsort(tian,n,sizeof(tian[0]),comp);
        qsort(qi,n,sizeof(qi[0]),comp);
        int lt=0,lq=0,rt=n-1,rq=n-1;
        while(lt<=rt)
        {
            if(tian[rt]>qi[rq])                    //如果田忌的最慢马比齐王的最慢马快，直
接比；
            {
                count++;
                rt--;
                rq--;
            }
            else if(tian[rt]<qi[rq])               //如果田忌的最慢马比齐王的最慢马慢，把齐
王最快马拉下水；
            {
                count--;
                lq++;
                rt--;
            }
            else if(tian[lt]>qi[lq])               //当田忌最慢马和齐王最慢马一样快时，如果
田忌最快马比齐王最快马快，把齐王最快马干掉
            {
                count++;
                lt++;
                lq++;
            }
            else
            {
                if(tian[lt]<qi[lq])                //当田忌最慢马和齐王最慢马一样快而且田
忌最快马比齐王最快马慢，田忌最慢马和齐王最快马比
                {
                    count--;
                    lq++;
                    rt--;
                }
                else            //当田忌最慢马和齐王最慢马一样快而且两者最快马一样
快时，田忌最慢马和齐王最快马比
                {
                    if(lt==rt||tian[rt]==qi[lq])    //如果检索已经到中间，或者田忌最慢的
马和齐王最快的马一样快
                        ;
                    else
                        count--;
                    rt--;
                    lq++;
                }
            }
        }
        printf("%d\n",count*200);
    }
    return 0;
}
int comp(const void *a,const void *b)
{
    return *(int *)b-*(int *)a;
}
```

73.弦截法求方程的根

```
#include<stdio.h>
#include<math.h>
double f(double x);
double xpoint(double x1,double x2);
double root(double x1,double x2);
int main(void)
{
	double x1,x2;
	double a;
	do
	{
		scanf("%lf%lf",&x1,&x2);
	}while(f(x1)*f(x2)>0);
	a=root(x1,x2);
	printf("%lf",a);
	return 0;
}
double f(double x)
{
	double y;
	y=x*x*x-5*x*x+16*x-80.0;
	return y;
}
double xpoint(double x1,double x2)
{
	double x;
	x=(x1*f(x2)-x2*f(x1))/(f(x2)-f(x1));
	return x;
}
double root(double x1,double x2)
{
	double x;
	do
	{
		x=xpoint(x1,x2);
		if(f(x)>0)
		{
			x2=x;
		}
		else
		{
			x1=x;
		}
	}while(fabs(f(x))>0.00001);
	return x;
}
```

74.填词

```
#include <stdio.h>
int main(void)
{
	int characters[26];
	int n, m, p; //输入的第一行，输入包括一个 n*m 的矩阵，和 p 个单词。
	int i, j; //循环变量
	for(i = 0; i < 26; i++) // 赋初值
		characters[i] = 0;
	scanf("%d%d%d", &n, &m, &p);
	for(i = 0; i < n; i++){//这一段读入 n*m 的矩阵，并记录矩阵中每个字母出现的次数
		char str[11];
```

```
            scanf("%s", str);
            for(j = 0; str[j] != '\0'; j++)
            characters[str[j] - 'A'] ++;
        }
        for(i = 0; i < p; i++){ //这一段读入 p 个单词，并且将单词中出现的字母在
//上一段的累计数组中去掉
             char str[200];
             scanf("%s", str);
             for(j = 0; str[j] != '\0'; j++)
             characters[str[j] - 'A'] --;
         }
        for(i = 0; i < 26; i++){ // 这一段输出所有出现次数大于 0 的字母。
            if(characters[i] != 0)
            for(j = 0; j < characters[i]; j++)
            printf("%c", i + 'A');
        }
        printf("\n");
        return 0;
}
```

75.棋盘上的距离

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int min(int a,int b)
{
    if(a>b)
        return b;
    else
        return a;
}
int main(void)
{
    int i,j,k,l,n;
    char x,y,z;
    scanf("%d",&n);
    while(n--)
    {
        scanf("%c%c%d %c%d",&z,&x,&i,&y,&j);
        k=abs(x-y);
        l=abs(i-j);
        printf("%d ",min(k,l)+abs(k-l));
        if(k==0&&l==0)
            printf("0 ");
        else if(k==0||l==0||k==l)
            printf("1 ");
        else
            printf("2 ");
        if(k==0&&l==0)
            printf("0 ");
        else if(k==0||l==0)
            printf("1 ");
        else
            printf("2 ");
        if(k==0&&l==0)
            printf("0\n");
        else if((k+l)%2==0&&k==l)
            printf("1\n");
        else if((k+l)%2==0&&k!=l)
```

```
            printf("2\n");
        else
            printf("Inf\n");
    }
    return 0;
}
```

76.装箱问题

```
#include <stdio.h>
int main(void)
{
    int N, a, b, c, d, e, f, y, x;//N 用来存储需要的箱子数目，y 用来存储 2*2 的空位数目
    // x 用来存储 1*1 的空位数目。
    int u[4]={0, 5, 3, 1};
    //数组 u 表示 3*3 的产品数目分别是 4 的倍数，4 的倍数+1, 4 的倍数+2, 4 的倍数+3
    //时，为 3*3 的产品打开的新箱子中剩余的 2*2 的空位的个数
    while(1){
    scanf("%d%d%d%d%d%d", &a, &b, &c, &d, &e, &f);
    if (a == 0 && b == 0 && c == 0 && d == 0 && e == 0 && f == 0) break;
    N = f + e + d + (c + 3) / 4;
    //这里有一个小技巧 - (c+3)/4 正好等于 c 除以 4 向上取整的结果,下同
    y = 5 * d + u[c % 4];
    if(b > y) N += (b - y + 8 ) / 9;
    x = 36 * N - 36 * f - 25 * e - 16 * d - 9 * c - 4 * b;
    if(a > x) N += ( a - x + 35 ) / 36;
    printf("%d\n", N);
    }
    return 0;
}
```

77.密码（Bob and Alice）

```
#include<stdio.h>
#include<string.h>
int main(void)
{
    int p,i,n,j,z=0,m,a[200],b[200];
    char c[200]={'\0'};
    char d[1000]={'\0'};
    while(scanf("%d",&n)&&n!=0)
    {
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
        while(1)
        {
            scanf("%d ",&p);
            if(p==0)
                break;
            for(i=0;i<n;i++){b[i]=i;c[i]='\0';}
            gets(c);
            m=strlen(c);
            for(m;m<n;m++)
            {
                c[m]=' ';
            }
            for(p;p>0;p--)
            {
                for(i=0;i<n;i++)
                {
                    for(j=0;j<n;j++)
                    {
```

```
                                if(a[j]==b[i]+1)
                                {
                                        b[i]=j;
                                        break;
                                }
                        }
                }
              }
             z++;
             for(i=0;i<n;i++)
                    d[i+n*(z-1)]=c[b[i]];
        }
           for(i=0;i<z;i++)
           {
                    for(j=(n*i);j<(n*(i+1));j++)
                    {
                          printf("%c",d[j]);
                    }
                    printf("\n");
           }
    }
    return 0;
}
```

78. skew 数

```
#include<stdio.h>
#include<string.h>
int main(void)
{
    int i,k,base[31],sum;
    char skew[32];
    base[0]=1;
    for( i = 1; i < 31; i++)
        base[i] = 2 * base[i-1] + 1;
    while(1) {
        scanf("%s", skew);
        if (strcmp(skew,"0") == 0)
        break;
        sum = 0;
        k = strlen(skew);
        for( i = 0; i < strlen(skew); i++) {
        k--;
        sum += (skew[i] - '0') * base[k];
        }
        printf("%d\n",sum);
        }
    return 0;
}
```

79.1082 子串

```
#include<stdio.h>
#include<string.h>
char string[100][101];
void strre(char *s)
{
    int len,i,j;
    len=strlen(s);
    char revs[len];
    for(i=0,j=len-1;j>=0;i++,j--)
    {
```

```
            revs[i]=s[j];
        }
        strcpy(s,revs);
}
int main(void)
{
        int t,n,m,i,j,k,minlen,len,flag,mins;
        char minstr[101];
        char revstr[101];
        scanf("%d",&t);
        while(t--)
        {
                scanf("%d",&n);
                minlen=110;
                for(m=0;m<n;m++)
                {
                        scanf("%s",string[m]);
                        len=strlen(string[m]);
                        if(len<minlen)
                        {
                                minlen=len;
                                mins=m;
                        }
                }
          for(i=minlen;i>=1;i--)
          {
            for(j=0;j<=minlen-i;j++)
            {
              strncpy(minstr,string[mins]+j,i);
              minstr[i]='\0';
              strcpy(revstr,minstr);
              strre(revstr);
              revstr[i]='\0';
              flag=1;
              for(k=0;k<n;k++){
                if(strstr(string[k],minstr)==NULL&&strstr(string[k],revstr)==NULL){
                  flag=0;
                  break;
                }
              }
              if(flag==1)
                goto OUTPUT;
            }
          }
        OUTPUT:
           printf("%d\n",i);
        }
        return 0;
}
```

80.POJ 2804 词典

```
#include<iostream>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
using namespace std;
class dic{
public:
        char engl[12];
```

```
    char fore[12];
};
int cmp(const void *a,const void *b){
    return strcmp(((dic *)a)->fore,((dic *)b)->fore);
}
int search(const void *a,const void *b){
    return strcmp((char *)a,((dic *)b)->fore);
}
int main(){
    dic words[100002];
    char word[12],ch,tmp[25];
    int count=0,i,j;
    dic *p;
    //读字典
    gets(tmp);
    while(tmp[0]!='\0'){
        for(i=0;tmp[i]!=' ';i++)
            words[count].engl[i]=tmp[i];
        words[count].engl[i]='\0';
        for(j=0,i++;tmp[i]!='\0';i++,j++)
            words[count].fore[j]=tmp[i];
        words[count].fore[j]='\0';
        count++;
        gets(tmp);
    }
    //getchar();
    qsort(words,count,sizeof(words[0]),cmp);
    //读文档
    while(cin>>word&&word){
        p=NULL;
        p=(dic *)(bsearch(word,words,count,sizeof(words[0]),search));
        if(p)
            cout<<p->engl<<endl;
        else
            cout<<"eh"<<endl;
    }
    return 0;
}
```

81.最短前缀

```
#include<stdio.h>
#include<string.h>
char word[1001][31];
char qz[1001][31];
int main(void)
{
    int i,j,k;
    int n=1,m;
    int flag,haha;
    while(scanf("%s",word[n++])==1)
        ;
    for(i=1;i<n;i++)       //给每个单词选最短前缀
    {
        haha=0;
        for(j=0;j<strlen(word[i]);j++)         //每个单词从头开始遍历
        {
            if(haha==1)
                break;
            qz[i][j]=word[i][j];
```

```
                if(j==strlen(word[i])-1)
                {
                    qz[i][j+1]='\0';
                    printf("%s %s\n",word[i],qz[i]);
                    haha=1;
                    break;
                }
                flag=0;
                for(k=1;k<n;k++)
                {
                    for(m=0;m<=j;m++)
                    {
                        if(qz[i][m]!=word[k][m]&&k!=i)
                        {
                            flag++;
                            break;
                        }

                    }
                    if(flag<k-1)
                        break;
                }
                if(flag==n-2)
                {
                    qz[i][j+1]='\0';
                    printf("%s %s\n",word[i],qz[i]);
                    haha=1;
                }
            }
        }
        return 0;
}
```

82.花生问题（多组）

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <math.h>
int T, M, N, K;
#define MAX_NUM 55
int aField[MAX_NUM][MAX_NUM];
int main()
{
scanf("%d", &T);
for( int t = 0; t < T; t ++ ) {
scanf("%d%d%d", &M, &N , &K);
//花生地的左上角对应的数组元素是 aField[1][1]，路的纵坐标是 0
for( int m = 1; m <= M; m ++ )
for( int n = 1 ; n <= N; n ++ )
scanf("%d", & aField[m][n]);
int nTotalPeanuts = 0; //摘到的花生总数
int nTotalTime = 0; //已经花去的时间
int nCuri = 0, nCurj; //当前位置坐标，
//nCuri 代表纵坐标，开始是在路上，所以初值为 0
while( nTotalTime < K ) { //如果还有时间
int nMax = 0, nMaxi, nMaxj; // 最大的花生数目，及其所处的位置
//下面这个循环寻找下一个最大花生数目及其位置
for( int i = 1; i <= M; i ++ ) {
for( int j = 1; j <= N ;j ++) {
```

```
if( nMax < aField[i][j]) {
nMax = aField[i][j];
nMaxi = i;
nMaxj = j;
}
}
}
if( nMax == 0 ) //地里已经没有花生了
break;
if( nCuri == 0)
nCurj = nMaxj; /* 如果当前位置是在路上，那么应走到横坐标
37. nMaxj 处，再进入花生地 */
/* 下一行看剩余时间是否足够走到 (nMaxi, nMaxj)处，摘取花生，
39. 并回到路上 */
if( nTotalTime + nMaxi + 1 + abs(nMaxi-nCuri) + abs(nMaxj-nCurj) <= K ) {
//下一行加上走到新位置，以及摘花生的时间
nTotalTime += 1 + abs(nMaxi-nCuri) + abs(nMaxj-nCurj);
nCuri = nMaxi; nCurj = nMaxj; //走到新的位置
nTotalPeanuts += aField[nMaxi][nMaxj];
aField[nMaxi][nMaxj] = 0; //摘走花生
}
else
break;
}
printf("%d\n", nTotalPeanuts);
}
}
```

83.排列

大家知道，给出正整数n，则1 到n 这n 个数可以构成n！种排列，把这些排列按照从小到大的顺序（字典顺序）列出，如n=3 时，列出1 2 3，1 3 2，2 1 3，2 3 1，3 1 2，3 2 1六个排列。

给出某个排列，求出这个排列的下k 个排列，如果遇到最后一个排列，则下1 排列为第1 个排列，即排列1 2 3…n。

比如：n = 3，k=2 给出排列2 3 1，则它的下1 个排列为3 1 2，下2 个排列为3 2 1，因此答案为 3 2 1。

这道题目，最直观的想法是求出 1 到n 的所有排列，然后将全部排列排序……且慢，n最大可以是1024，1024! 个排列，几乎永远也算不出来，算出来也没有地方存放。那么，有没有公式或规律，能够很快由一个排列推算出下k 个排列呢？实际上寻找规律或公式都是徒劳的，只能老老实实由给定排列算出下一个排列，再算出下一个排列……一直算到第k的排列。鉴于k 的值很小，最多只有64，因此这种算法应该是可行的。

如何由给定排列求下一个排列？不妨自己动手做一下。比如：

“2 1 4 7 6 3 5”的下一个排列是什么？容易，显然是“2 1 4 7 6 5 3”，那么，再下一个排列是什么？有点难了，是“2 1 5 3 4 6 7”。

以从“2 1 4 7 6 5 3”求出下一个排列 “2 1 5 3 4 6 7”作为例子，可以总结出求给定排列的下一个排列的步骤：

假设给定排列中的n 个数从左到右是a1, a2, a3……an 。

1) 从 an 开始，往左边找，直到找到某个aj，满足aj-1 < aj（对上例, 这个aj 就是 7，aj-1 就是4）。

2) 在 aj 、aj+1…… an 中找到最小的比aj-1 大的数，将这个数和 aj-1 互换位置（对上例, 这个数就是5，和4 换完位置后的排列是 “2 1 5 7 6 4 3”）。

3) 将从位置 j 到位置n 的所有数（共n-j+1 个）从小到大重新排序，排好序后，新的排列就是所要求的排列。（对上例，就是将“7 6 4 3”排序，排好后的新排列就是“2 1 5 3 4 6 7”）。

当然，按照题目要求，如果a1, a2, a3……an 已经是降序，那么它的下一个排序就是an, an-1, an-2……a1 。

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_NUM 1024
```

```
int an[MAX_NUM + 10];
//用以排序的比较函数
int MyCompare( const void * e1, const void * e2)
{
return * ((int *) e1) - * ((int *) e2);
}
main()
{
int M;
int n, k, i, j;
scanf("%d", & M);
for (int m = 0; m < M; m ++ ) {
scanf("%d%d", &n, &k);
//排列存放在  an[1] .... an[n]
for( i = 1; i <= n; i ++ )
scanf("%d", &an[i]);
an[0] = 100000; //确保 an[0]比排列中所有的数都大
for( i = 0; i < k ;i ++ ) { //每次循环都找出下一个排列
for( j = n ; j >= 1 && an[j-1] > an[j] ; j -- ) ;
if( j >= 1 ) {
int nMinLarger = an[j];
int nMinIdx = j;
//下面找出从 an[j]及其后最小的比  an[j-1]大的元素，并记住其下标
for( int kk = j; kk <= n; kk ++)
if( nMinLarger > an[kk] && an[kk] > an[j-1]) {
nMinLarger = an[kk];
nMinIdx = kk;
}
//交换位置
an[nMinIdx] = an[j-1];
an[j-1] = nMinLarger;
qsort( an + j, n - j + 1 , sizeof(int), MyCompare); //排序
}
else { //an  里的排列已经是降序了，那么下一个排列就是  1 2 3  。。。。  n
for( j = 1; j <= n; j ++ )
an[j] = j;
}
}
for( j = 1; j <= n; j ++ )
printf("%d ", an[j]);
printf("\n");
}
}
```

84.循环数（Round and Round We Go）

```
#include<stdio.h>
#include<string.h>
char s[100];          //用来存储读入的数
int s1[100];
int s2[100];
int main(void)
{
    int i,j,k,wei,flag,m,count,x;
    while(scanf("%s",s)==1)
    {
        count=0;
        wei=strlen(s);
        memset(s1,0,sizeof(s1));
        for(i=0,j=wei-1;i<strlen(s);i++)
```

```
        {
            s1[j]=s[i]-'0';
            s2[j]=s[i]-'0';
            j--;
        }
        for(i=1;i<=wei;i++)
        {
            for(j=0;j<wei;j++)
                s1[j]=s2[j]*i;
            for(j=0;j<wei;j++)
            {
                while(s1[j]>=10)
                {
                    s1[j]-=10;
                    s1[j+1]+=1;
                }
            }
            flag=0;
            x=0;
            for(j=0;j<wei;j++)
            {
                m=j;
                for(k=wei-1;k>=0;k--)
                {
                    if(s1[k]!=s[m%wei]-'0')
                        break;
                    else
                    {
                        m++;
                    }
                }
                if(m==j+wei)
                {
                    x=1;
                    break;
                }
            }
            if(x==1)
                count++;
        }
        if(count==wei)
            printf("%s is cyclic\n",s);
        else
            printf("%s is not cyclic\n",s);
    }
    return 0;
}
```

85.麦森数

```
#include <stdio.h>
#include <memory.h>
#define LEN 125 //每数组元素存放十进制的 4 位，因此数组最多只要 125 个元素即可。
#include <math.h>
/* Multiply 函数功能是计算高精度乘法 a * b
6. 结果的末 500 位放在 a 中
7. */
void Multiply(int* a, int* b)
{
    int i, j;
```

```
        int nCarry; //存放进位
        int nTmp;
        int c[LEN]; //存放结果的末 500 位
        memset(c, 0, sizeof(int) * LEN);
        for (i=0;i<LEN;i++) {
        nCarry=0;
        for (j=0;j<LEN-i;j++) {
        nTmp=c[i+j]+a[j]*b[i]+nCarry;
        c[i+j]=nTmp%10000;
        nCarry=nTmp/10000;
        }
        }
        memcpy( a, c, LEN*sizeof(int));
}
int main()
{
        int i;
        int p;
        int anPow[LEN]; //存放不断增长的 2 的次幂
        int aResult[LEN]; //存放最终结果的末 500 位
        scanf("%d", & p);
        printf("%d\n", (int)(p*log10(2))+1);
        //下面将 2 的次幂初始化为 2^(2^0)(a^b 表示 a 的 b 次方)，
        // 最终结果初始化为 1
        anPow[0]=2;
        aResult[0]=1;
        for (i=1;i<LEN;i++) {
        anPow[i]=0;
        aResult[i]=0;
}
//下面计算 2 的 p 次方
        while (p>0) { // p = 0 则说明 p 中的有效位都用过了，不需再算下去
        if ( p & 1 ) //判断此时 p 中最低位是否为 1
        Multiply(aResult, anPow);
        p>>=1;
        Multiply(anPow, anPow);
        }
        aResult[0]--; //2 的 p 次方算出后减 1

//输出结果
        for (i=LEN-1;i>=0;i--) {
        if (i%25==12)
        printf("%02d\n%02d", aResult[i]/100,
        aResult[i]%100);
        else {
        printf("%04d", aResult[i]);
        if (i%25==0)
        printf("\n");
        }
        }
        return 0;
}
86.浮点数加法
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define LEN 101
int ia[LEN], fa[LEN], ib[LEN], fb[LEN], ic[LEN], fc[LEN];
```

```
int main()
{
	char str1[LEN], str2[LEN];
	int i, j, k, n, l1, l2, lai, laf, lbi, lbf, temp, flmax, ilmax;
	while (scanf("%d", &n) != EOF) {
		while (n --) {
			// 初始化
			memset(ia, 0, sizeof(ia));
			memset(fa, 0, sizeof(fa));
			memset(ib, 0, sizeof(ib));
			memset(fb, 0, sizeof(fb));
			memset(ic, 0, sizeof(ic));
			memset(fc, 0, sizeof(fc));
			// 接收第一个浮点数
			scanf("%s", str1);
			l1 = strlen(str1);
			// 构建整数部分数组
			for (i = 0; i < l1 && str1[i] != '.'; i ++) {
				ia[i] = str1[i] - '0';
			}
			lai = i;
			// 数位替换
			for (j = 0, k = lai - 1; j <= lai / 2 && j < k; j ++, k --) {
				temp = ia[j];
				ia[j] = ia[k];
				ia[k] = temp;
			}
			// 构建小数部分数组
			for (i += 1; i < l1; i ++) {
				fa[i - 1 - lai] = str1[i] - '0';
			}
			laf = i - 1 - lai;
			// 接收第二个浮点数
			scanf("%s", str2);
			l2 = strlen(str2);
			// 构建整数部分数组
			for (i = 0; i < l2 && str2[i] != '.'; i ++) {
				ib[i] = str2[i] - '0';
			}
			lbi = i;
			// 数位替换
			for (j = 0, k = lbi - 1; j <= lbi / 2 && j < k; j ++, k --) {
				temp = ib[j];
				ib[j] = ib[k];
				ib[k] = temp;
			}
			// 构建小数部分数组
			for (i += 1; i < l2; i ++) {
				fb[i - 1 - lbi] = str2[i] - '0';
			}
			lbf = i - 1 - lbi;
			// 谁的小数位数更多
			flmax = (laf >= lbf) ? laf : lbf;
			int c = 0;	//小数进位
			for (i = 0, j = flmax - 1; j >= 0; j --, i ++) {
				fc[i] = fa[j] + fb[j] + c;
				if (fc[i] >= 10) {
					c = fc[i] / 10;
```

```
                        fc[i] %= 10;
                    }else {
                        c = 0;
                    }
                }
                // 整数相加
                ilmax = (lai >= lbi) ? lai : lbi;
                for (i = 0; i < ilmax; i ++) {
                    ic[i] = ia[i] + ib[i] + c;
                    if (ic[i] >= 10) {
                        c = ic[i] / 10;
                        ic[i] %= 10;
                    }else {
                        c = 0;
                    }
                }
                while (c) {
                    ic[ilmax ++] = c % 10;
                    c /= 10;
                }
                // 打印最后结果
                // 找到第一个不为 0 的整数位
                for (j = ilmax - 1; j >= 0; j --) {
                    if (ic[j] != 0) {
                        break;
                    }
                }
                if (j >= 0) {
                    for (i = j; i >= 0; i --) {
                        printf("%d", ic[i]);
                    }
                }else {
                    printf("0");
                }
                printf(".");
                // 找到最后一个不为 0 的小数位
                for (j = 0; j < flmax - 1; j ++) {
                    if (fc[j] != 0) {
                        break;
                    }
                }
                for (i = flmax - 1; i >= j; i --) {
                    printf("%d", fc[i]);
                }
                printf("\n");
                // 接收空行
                getchar();
            }
        }
        return 0;
}
```

87.浮点数高精度幂

```
#include <iostream>
#include <string.h>
#include <memory.h>
#include <stdio.h>
using namespace std;
#define MAX_LEN 500
```

```
unsigned an1[MAX_LEN * 2 +10];
unsigned an2[MAX_LEN+10];
unsigned aResult[MAX_LEN * 2 + 10];
int Multiply( unsigned * a1,int nLen1, unsigned * a2, int nLen2, unsigned * ar)
// ar= a1 * a2, a1 长度 nLen1, a2  长度 nLen2,返回结果的长度
{
	int i,j;
	memset( ar, 0, sizeof(aResult));
	for( i= 0;i < nLen2; i++ ) {
//每一轮都用 a1 的一位，去和 a2 各位相乘,从 a1 的个位开始
	for( j = 0; j < nLen1; j ++ ) //用选定的 a1 的那一位，去乘
//a2 的各位
	ar[i+j] += a2[i]*a1[j]; //两数第 i, j 位相乘，累加到结
//果的第 i+j 位
}
	int nLen= 0;
	//下面的循环统一处理进位问题
	for( i= 0; i< MAX_LEN * 2; i++ )
	{
	if( ar[i] >= 10 )
	{
		ar[i+1] += ar[i] / 10;
		ar[i] %= 10;
	}
	if( ar[i] )
	nLen= i+ 1;
	}
	return nLen;
}
int main(void)
{
	char R[30];
	int n;
	int nBits;
	int i;
	while( scanf("%s%d" ,R,&n)!= EOF)
	{
	int nLen= strlen(R);
	int j=0;
	nBits= 0;
	memset(an1,0,sizeof(an1));
	memset(an2,0,sizeof(an2));
	bool bDecMeet= false;
	bool bNoneZeroMeet= false;
	//是否碰到小数点了 boolbNoneZeroMeet= false;
	  //是否碰到非 0 位了
//下面把 R 去掉小数点变成高精度表示形式，并算小数点后位数
for(j=0,i = nLen-1; i>= 0; i--) {
if( R[i] == '.') {
bNoneZeroMeet= true;
bDecMeet= true;
}
else {
if( R[i] != '0' ) bNoneZeroMeet= true;
if( bNoneZeroMeet) {
an1[j++] = R[i] -'0';
if(!bDecMeet) nBits++;
}
```

```
}
}
if( !bDecMeet)
nBits= 0;
int nLen2; //R 去掉小数点变成整数后的长度
if( bDecMeet) {
if( nBits)
nLen2 = nLen-1;
else
nLen2 = strchr(R,'.') -R;
}
else
nLen2 = nLen;
int nLen1 = nLen2; //结果长度
memcpy(an2,an1,sizeof(an2));
memcpy(aResult,an1,sizeof(an1));
for( i= 0; i< n -1;i ++ ) {
nLen1 = Multiply(an1,nLen1, an2,nLen2, aResult);
memcpy( an1,aResult, sizeof(aResult));
}
int nTotalBits= nBits* n ;
//结果的小数点后面的位数
//下面输出结果
if( nLen1 == 0)
//结果就是 0
printf("0");
else if( nLen1 <= nTotalBits)
{//结果小于 1
printf(".");
int i;
for( i= 0;i < nTotalBits-nLen1;i++)
printf("0");
for( i= nLen1-1; i>=0; i--)
printf("%d",aResult[i]);
}
else
{
for( int i= nLen1-1; i>=0; i--)
{
printf("%d",aResult[i]);
if( nTotalBits&& i== nTotalBits)
printf(".");
}
}
printf("\n");
}
return 0;}
```

88.硬币面值

```
#include<stdio.h>
int a[101];
int c[101];
int dp[100001]={0};//dp[m]==1 表示 m 能取到
int num[100001];     //num[j]表示在增加第 i 种硬币时，取到 dp[j]所需要的这种硬币的个数，
不能多于 c[i]
int main(void)
{
        int i,j,k,temp;
        int n,m;
```

```
	int count=0;
	scanf("%d%d",&n,&m);
	for(i=1;i<=n;i++)
		scanf("%d",&a[i]);
	for(i=1;i<=n;i++)
		scanf("%d",&c[i]);
	dp[0]=1;
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=m;j++)
			num[j]=0;
		for(j=1;j<=m;j++)
		{
			if(j-a[i]>=0&&dp[j]==0)
			{
				if(dp[j-a[i]]==1&&num[j-a[i]]<c[i])
				{
					dp[j]=1;
					num[j]=num[j-a[i]]+1;
				}
			}
		}
	}
	int sum=0;
	for(i=1;i<=m;i++)
	{
		if(dp[i]==1)
			sum++;
	}
	printf("%d",sum);
	return 0;
}
```

89.走出迷宫（习题 15-6）

```
#include<stdio.h>
char s[101][101];
int dp[101][101];
int n,m;
int min=0;
int x,y,x1,y1;
void f(int i,int j)
{
	int t=dp[i][j];
	if(i==x1&&j==y1)
		min=t;
	t++;
	if(i+1<n&&s[i+1][j]!='#'&&dp[i+1][j]>t)
	{
		dp[i+1][j]=t;
		f(i+1,j);
	}
	if(j+1<m&&s[i][j+1]!='#'&&dp[i][j+1]>t)
	{
		dp[i][j+1]=t;
		f(i,j+1);
	}
	if(i>0&&s[i-1][j]!='#'&&dp[i-1][j]>t)
	{
		dp[i-1][j]=t;
```

```
            f(i-1,j);
        }
        if(j>0&&s[i][j-1]!='#'&&dp[i][j-1]>t)
        {
            dp[i][j-1]=t;
            f(i,j-1);
        }
}
int main(void)
{
        int i,j,k;
        scanf("%d%d",&n,&m);
        for(i=0;i<n;i++)
            scanf("%s",s[i]);
        for(i=0;i<n;i++)
            for(j=0;j<m;j++)
                dp[i][j]=10000;
        for(i=0;i<n;i++)
        {
            for(j=0;j<m;j++)
            {
                if(s[i][j]=='S')
                {
                    x=i;
                    y=j;
                }
                if(s[i][j]=='T')
                {
                    x1=i;
                    y1=j;
                }
            }
        }
        dp[x][y]=0;
        f(x,y);
        printf("%d",min);
        return 0;
}
//BFS
#include<stdio.h>
#include<iostream>
using namespace std;
#include<queue>
char s[101][101];
int dp[101][101]={0};
struct node
{
        int x;
        int y;
        int t;
}start;
int row,col,x1,y1,x2,y2,k,min;
int bfs(node s)
{
        queue<node> q;                                  //定义一个队列
        node a,b;
        int i;
        int dir[4][2]={{0,1},{0,-1},{1,0},{-1,0}};
```

```
        q.push(s);                                          //压入队列
        dp[s.x][s.y]=1;                                     //表示已经走过
        while(!q.empty())                                   //当队列非空时
        {
            a=q.front();                                    //取出首元素
            q.pop();                                        //删除首元素
            if(a.x==x2&&a.y==y2)
                return a.t;
            for(i=0;i<4;i++)
            {
                int x=a.x+dir[i][0];
                int y=a.y+dir[i][1];
                int t=a.t+1;
                if(x>=0&&x<col&&y>=0&&y<row&&dp[x][y]==0)
                {
                    b.x=x;
                    b.y=y;
                    b.t=t;
                    q.push(b);
                    dp[b.x][b.y]=1;
                }
            }
        }
}
int main(void)
{
        int i,j,k;
        scanf("%d%d",&row,&col);
        for(i=0;i<row;i++)
            scanf("%s",s[i]);
        for(i=0;i<row;i++)
            for(j=0;j<col;j++)
            {
                if(s[i][j]=='S')
                {
                    start.x=i;
                    start.y=j;
                    start.t=0;
                }
                if(s[i][j]=='T')
                {
                    x2=i;
                    y2=j;
                }
            }
        for(i=0;i<row;i++)
            for(j=0;j<col;j++)
                {
                    if(s[i][j]=='#')
                        dp[i][j]=100000;
                }
        printf("%d",bfs(start));
        return 0;
}
```

90.城堡（习题 15-7）

```
#include<stdio.h>
int m,n;
int a[100][100];
```

```
int dp[100][100]={0};
int sum[100];
int count=0;          //统计房间数
int max=0;             //记录最大房间
void f(int i,int j)
{
      dp[i][j]=1;
      if(a[i][j]>=8&&dp[i+1][j]==0&&i+1<m)
      {
            sum[count]++;
            //a[i][j]-=8;
            f(i+1,j);
      }
      if(a[i][j]%8>=4&&dp[i][j+1]==0&&j+1<n)
      {
            sum[count]++;
            //a[i][j]-=4;
            f(i,j+1);
      }
      if(a[i][j]!=0&&a[i][j]!=1&&a[i][j]!=4&&a[i][j]!=5&&a[i][j]!=8&&a[i][j]!=9&&a[i][j]!=12
&&a[i][j]!=13&&dp[i-1][j]==0&&i>0)
      {
            sum[count]++;
            //a[i][j]-=2;
            f(i-1,j);
      }
      if(a[i][j]%2==1&&dp[i][j-1]==0&&j>0)
      {
            sum[count]++;
            f(i,j-1);
      }
}
int main(void)
{
      //freopen("abc.in","r",stdin);
      int i,j,k;
      scanf("%d%d",&m,&n);
      for(i=0;i<100;i++)
            sum[i]=1;
      for(i=0;i<m;i++)
      {
            for(j=0;j<n;j++)
            {
                  scanf("%d",&a[i][j]);
                  a[i][j]=15-a[i][j];
            }
      }
      for(i=0;i<m;i++)
      {
            for(j=0;j<n;j++)
            {
                  if(dp[i][j]==0)
                  {
                        f(i,j);
                        if(sum[count]>max)
                              max=sum[count];
                        count++;
                  }
```

```
            }
        }
        printf("%d\n",count);
        printf("%d",max);
        return 0;
}
```

91.生理周期

```
#include <stdio.h>
int main(void)
{
        int p,e,i,d,j,no=1;
            scanf("%d%d%d%d", &p, &e, &i, &d);
        while(p!=-1 && e!=-1 && i!=-1 && d!=-1)
        {
            for(j=d+1; j<21252; j++)
                if ((j-p)%23 == 0) break;
            for( ; j<21252; j=j+23)
                if ((j-e)%28 == 0) break;
            for( ; j<21252; j=j+23*28)
                if ((j-i)%33 == 0) break;
            printf("Case %d", no);
            printf(": the next triple peak occurs in %d days.\n", j-d);
            scanf("%d%d%d%d", &p, &e, &i, &d);
            no++;
        }
        return 0;
}
```

92.熄灯问题

```
#include <stdio.h>
int puzzle[6][8], press[6][8];
bool guess()
{
        int c, r;
        for ( r = 1; r < 5; r++ )
        for ( c = 1; c < 7; c++ )
        press[r+1][c] =(puzzle[r][c] + press[r][c] + press[r-1][c] + press[r][c-1] + press[r][c+1]) % 2;
        for(c=1; c<7; c++)
        if ( (press[5][c-1] + press[5][c] + press[5][c+1] + press[4][c]) % 2 != puzzle[5][c] )
        return(false);
        return( true );
}
void enumate( )
{
        int c;
        bool success;
        for ( c = 1; c < 7; c++)
        press[1][c] = 0;
        while( guess() == false ) {
            press[1][1]++;
            c = 1;
            while ( press[1][c] > 1 ) {
            press[1][c] = 0;
            c++;
            press[1][c]++;
            }
        }
}
int main(void)
```

```
{
	int cases, i, r, c;
	scanf("%d", &cases);
	for ( r = 0; r < 6; r++ )
	press[r][0] = press[r][7] = 0;
	for ( c = 1; r < 7; r++ )
		press[0][c] = 0;
	for ( i = 0; i < cases; i++ ) {
		for ( r = 1; r < 6; r++ )
		for ( c = 1; c < 7; c++ )
		scanf("%d", &puzzle[r][c]);
		enumate( );
		printf("PUZZLE #%d\n", i+1);
		for ( r = 1; r < 6; r++ ) {
			for ( c = 1; c < 6; c++ )
				printf("%d ",press[r][c]);
			printf("%d\n",press[r][c]);
		}
	}
	return 0;
}
```

93.恼人的青蛙

```
#include <stdio.h>
#include <stdlib.h>
int r, c, n;
struct PLANT
{
	int x, y;
};
PLANT plants[5001];
PLANT plant;
int myCompare( const void *ele1, const void *ele2 );
int searchPath(PLANT secPlant, int dX, int dY);
int main(void)
{
	int i, j, dX, dY, pX, pY, steps, max = 2;
	scanf("%d%d", &r, &c);
	scanf("%d", &n);
	for ( i = 0; i < n; i++ )
		scanf("%d %d", &plants[i].x, &plants[i].y);
	qsort(plants, n, sizeof(PLANT), myCompare);
	for ( i = 0; i < n - 2; i++ )
		for ( j = i + 1; j < n - 1; j++ )
		{
			dX = plants[j].x - plants[i].x;
			dY = plants[j].y - plants[i].y;
			pX = plants[i].x - dX;
			pY = plants[i].y - dY;
			if ( pX <= r && pX >= 1 && pY <= c && pY >= 1 )
				continue;
			if ( plants[i].x + max * dX > r )
				break;
			pY = plants[i].y + max * dY;
			if ( pY > c || pY < 1)
				continue;
			steps = searchPath( plants[j], dX, dY );
			if ( steps > max ) max = steps;
		}
```

```
    if ( max == 2 ) max = 0;
        printf("%d\n", max);
    return 0;
}
int myCompare( const void *ele1, const void *ele2 )
{
    PLANT *p1, *p2;
    p1 = (PLANT*) ele1;
    p2 = (PLANT*) ele2;
    if ( p1->x == p2->x ) return( p1->y - p2->y );
    return ( p1->x - p2->x );
}
int searchPath( PLANT secPlant, int dX, int dY )
{
    PLANT plant;
    int steps;
    plant.x = secPlant.x + dX;
    plant.y = secPlant.y + dY;
    steps = 2;
    while ( plant.x <= r && plant.x >= 1 && plant.y <= c && plant.y >= 1 )
    {
        if ( !bsearch(&plant, plants, n, sizeof(PLANT), myCompare) )
        {
            steps = 0;
            break;
        }
        plant.x += dX;
        plant.y += dY;
        steps++;
    }
    return( steps );
}
```

94.poj1160 建邮局 山区建小学（习题 15-8）

```
#include<stdio.h>
int dp[1000][1000]={0};                    //dp[m][n]表示 m 个村庄修建 n 个学校的最短距离和
int l[1000][1000]={0};                     //l[k+1][m]表示第 k+1 个村庄到第 m 个村庄只修一个学校的最短距离
int dis[1000];
int dd[1000];                              //存储村庄之间的距离
int main(void)
{
    int i,j,k,m,n,a,b;
    //freopen("in.txt","r",stdin);
    scanf("%d%d",&m,&n);        //m 表示村庄个数，n 表示学校个数
    for(i=1;i<=m;i++)
        scanf("%d",&dd[i]);
    for(i=1;i<m;i++)
        for(j=i+1;j<=m;j++)
            l[i][j]=l[i][j-1]+(dd[j]-dd[(i+j)/2]);
    for(i=1;i<=m;i++)
        dp[i][1]=l[1][i];
        int len;
    for(i=1;i<=m;i++)
    {
        for(j=2;j<=n&&j<=i;j++)
        {
            int min=100000;
```

```
                for(k=j-1;k<=i-1;k++)
                {
                    if(j-1!=0)
                        len=dp[k][j-1]+l[k+1][i];
                    else
                        len=l[k][i];
                    if(len<min)
                        min=len;
                }
                dp[i][j]=min;
            }
        }
        printf("%d",dp[m][n]);
        return 0;
}
#include<stdio.h>
int dp[1000][1000]={0};                    //dp[m][n]表示 m 个村庄修建 n 个学校的最短距离
和
int l[1000][1000]={0};                     //l[k+1][m]表示第 k+1 个村庄到第 m 个村庄只修
一个学校的最短距离
int dis[1000];                             //存储村庄之间的距离
int dd[1000];
int main(void)
{
    int i,j,k,m,n,a,b;
    //freopen("in.txt","r",stdin);
    scanf("%d%d",&m,&n);        //m 表示村庄个数，n 表示学校个数
    for(i=1;i<m;i++)
        scanf("%d",&dis[i]);
    dd[1]=0;
    for(i=2;i<=m;i++)
        dd[i]=dd[i-1]+dis[i-1];
    for(i=1;i<m;i++)
        for(j=i+1;j<=m;j++)
            l[i][j]=l[i][j-1]+(dd[j]-dd[(i+j)/2]);
    for(i=1;i<=m;i++)
        dp[i][1]=l[1][i];
    int len;
    for(i=1;i<=m;i++)
    {
        for(j=2;j<=n&&j<=i;j++)
        {
            int min=100000;
            for(k=j-1;k<=i-1;k++)
            {
                if(j-1!=0)
                    len=dp[k][j-1]+l[k+1][i];
                else
                    len=l[k][i];
                if(len<min)
                    min=len;
            }
            dp[i][j]=min;
        }
    }
    printf("%d",dp[m][n]);
    return 0;
}
```

95.计算对数

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include<stdlib.h>
int main(void)
{
	//freopen("in.txt","r",stdin);
	int i,j,n,a1,b1;
	char a[100],b[100];
	scanf("%d",&n);
	while(n--)
	{
		memset(a,'0',sizeof(a));
		memset(b,'0',sizeof(b));
		scanf("%s%s",a,b);
		a1=strlen(a)-1;
		b1=strlen(b)-1;
		a[6]=b[6]='\0';
		for(i=0;i<6;i++)
		{
			if(a[i]=='\0')
				a[i]='0';
			if(b[i]=='\0')
				b[i]='0';
		}
		printf("%d\n",(int)((b1+log10(atof(b)/1e5))/(a1+log10(atof(a)/1e5))));
	}
	return 0;
}
```

96.数字方格

```
#include<stdio.h>
int main(void)
{
	int t,x;
	scanf("%d",&t);
	while(t--)
	{
		int x,l,m,n,ok;
		scanf("%d",&x);
		ok=0;
		for(n=60;n>=1;n--)
		{
			for(m=66;m>=0;m--)
			{
				if(5*n-3*m>=0&&5*n-3*m<=x)
				{
					for(l=100;l>=0;l--)
					{

	if(2*l+3*m-5*n>=0&&2*l+3*m-5*n<=x&&5*n-2*l>=0&&5*n-2*l<=x)
						{
							ok=1;
							break;
						}
					}
				}
				if(ok==1)
```

```
                                        break;
                        }
                        if(ok==1)
                                        break;
                }
                printf("%d\n",5*n);
        }
}
```

97. Doomsday

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
char s[1000];
int a[100]={0};
int b[100]={0};
int ok=0;
int max;
void f(int i)
{
        if(i==max&&b[i]%3==0)
                ok=1;
        else if(b[i]%3==1&&b[i]>=0)
        {
                b[i]-=1;
                b[i+1]-=1;
                b[i+2]-=1;
                f(i+1);
        }
        else if(b[i]%3==2&&b[i]>=0)
        {
                b[i]-=2;
                b[i+1]-=2;
                b[i+2]-=2;
                f(i+1);
        }
        else if(b[i]%3==0&&b[i]>=0)
                f(i+1);
}
int main(void)
{
        int i=0,j,k,l,num,sum;
        char c;
        int wei;
        while(1)
        {
                gets(s);
                wei=strlen(s);
                for(j=0;j<wei;j++)
                {
                        if(isdigit(s[j]))
                        {
                                i++;
                                num=s[j]-'0';
                                if(num==0)
                                break;
                                if(num==1)      a[1]++;
                                if(num==2)      a[2]++;
                                if(num==3)      a[3]++;
```

```
                    if(num==4)      a[4]++;
                    if(num==5)      a[5]++;
                    if(num==6)      a[6]++;
                    if(num==7)      a[7]++;
                    if(num==8)      a[8]++;
                    if(num==9)      a[9]++;
                }
            }
            sum=0;
            if(num==0)
                break;
            if((i-2)%3!=0)
            {
                printf("XIANGGONG\n");
            }
            else
            {
                for(j=1;j<=9;j++)
                {
                    if(a[j]>=2)
                    {
                        sum=0;
                        for(k=1;k<=9;k++)
                        {
                            b[k]=a[k];
                            if(a[k]>0)
                                max=k;
                            for(l=1;l<=a[k];l++)
                                sum+=k;
                        }
                        sum-=2*j;
                        if(sum%3==0)
                        {
                            b[j]-=2;
                            f(1);
                            if(ok)
                            {
                                printf("HU\n");
                                ok=0;
                                break;
                            }
                        }
                    }

                    for(k=1;k<=9;k++)
                        b[k]=a[k];
                }
                if(j==10)
                {
                    printf("BUHU\n");
                }
            }
            memset(a,0,sizeof(a));
            i=0;
        }
        return 0;
}
```

98. Saturday

```
#include<stdio.h>
int main(void)
{
	int i,j,k,n;
	int h[100],m[100],s[100],y[100],M[100],d[100];
	int zao=10,wan=20;
	scanf("%d",&n);
	h[n+1]=0;m[n+1]=0;s[n+1]=0;y[n+1]=0;M[n+1]=0;d[n+1]=0;
	while(n--)
	{
		scanf("%d%d%d%d%d%d",&h[n],&m[n],&s[n],&y[n],&M[n],&d[n]);
		if(y[n]==y[n+1]&&M[n]==M[n+1]&&d[n]==d[n+1])
		{

	if(h[n]==7&&m[n]>=0&&m[n]<=59&&s[n]>=0&&s[n]<=59||h[n]==8&&m[n]>=0&&m[n]
<=29&&s[n]>=0&&s[n]<=59||h[n]==8&&m[n]==30&&s[n]==0)
			;

	if(h[n]>=16&&h[n]<=20&&m[n]>=0&&m[n]<=59&&s[n]>=0&&s[n]<=59||h[n]==21&&m
[n]>=0&&m[n]<=29&&s[n]>=0&&s[n]<=59||h[n]==21&&m[n]==30&&s[n]==0)
		{

	if(h[n+1]>=16&&h[n+1]<=20&&m[n+1]>=0&&m[n+1]<=59&&s[n+1]>=0&&s[n+1]<=59|
|h[n+1]==21&&m[n+1]>=0&&m[n+1]<=29&&s[n+1]>=0&&s[n+1]<=59||h[n+1]==21&&m[n+1
]==30&&s[n+1]==0)

	if(h[n]-h[n+1]>=1||m[n]-m[n+1]>=31||(m[n]-m[n+1]==30&&s[n]-s[n+1]>=0))
				wan--;
		}
		}
		else
		{

	if(h[n]==7&&m[n]>=0&&m[n]<=59&&s[n]>=0&&s[n]<=59||h[n]==8&&m[n]>=0&&m[n]
<=29&&s[n]>=0&&s[n]<=59||h[n]==8&&m[n]==30&&s[n]==0)
			zao--;
		}
	}
	if(zao>=0&&wan>=0)
		printf("%d %d",zao,wan);
	if(zao<0&&wan<0)
		printf("%d %d",0,0);
	if(zao>=0&&wan<0)
		printf("%d %d",zao,0);
	if(zao<0&&wan>=0)
		printf("%d %d",0,wan);
	return 0;
}
```

99.分解因数

```
#include<stdio.h>
int sum;
void f(int i,int j)
{
	int k;
	if(i==1)
		sum++;
	else
		for(k=j;k<=i;k++)
```

```
                if(i%k==0)
                        f(i/k,k);
}
int main(void)
{
        int i,j,k;
        int n;
        int a;
        scanf("%d",&n);
        while(n--)
        {
                sum=1;
                scanf("%d",&a);
                for(i=2;i*i<=a;i++)
                {
                        if(a%i==0)
                                f(a/i,i);
                }
                printf("%d\n",sum);
        }
        return 0;
}
```

100.红与黑

```
#include<stdio.h>
#include<string.h>
char a[21][21];
int dp[21][21];
int x,y;
int count;
void f(int i,int j)
{
        dp[i][j]=1;
        if(i>0&&dp[i-1][j]==0&&a[i-1][j]=='.')
        {
                count++;
                f(i-1,j);
        }
        if(j>0&&dp[i][j-1]==0&&a[i][j-1]=='.')
        {
                count++;
                f(i,j-1);
        }
        if(i+1<x&&dp[i+1][j]==0&&a[i+1][j]=='.')
        {
                count++;
                f(i+1,j);
        }
        if(j+1<y&&dp[i][j+1]==0&&a[i][j+1]=='.')
        {
                count++;
                f(i,j+1);
        }
}
int main(void)
{
        //freopen("in.txt","r",stdin);
        int i,j,k;
        int x1,y1;
```

```
    while(1)
    {
        scanf("%d%d",&y,&x);
        count=1;
        for(i=0;i<=x;i++)
            for(j=0;j<y;j++)
                dp[i][j]=0;
        if(x==0)
            break;
        for(i=0;i<x;i++)
            scanf("%s",a[i]);
        for(i=0;i<x;i++)
            for(j=0;j<y;j++)
                if(a[i][j]=='@')
                {
                    x1=i;
                    y1=j;
                    break;
                }
        f(x1,y1);
        printf("%d\n",count);
    }
    return 0;
}
```

101.棋盘问题

```
#include<stdio.h>
#include<string.h>
char s[10][10];
int lie[10];                    //用来记录每一列是否放有其他棋子
int c;                          //记录放法总数
int n,k;
void dfs(int i,int num)
{
    int j,h;
    for(j=0;j<n;j++)                    //从第 0 列到第 n-1 列
    {
        if(s[i][j]=='#'&&lie[j]==0)
        {
            if(num==1)                  //如果只剩一枚棋子了，总数+1
                c++;
            else
            {
                lie[j]=1;
                for(h=i+1;h<=n-num+1;h++)       //不然，从下一行开始要放 num-1 个棋
子，同样最低从 n-num+1 行开始放
                    dfs(h,num-1);
                lie[j]=0;                       //放好之后清零
            }
        }
    }
}
int main(void)
{
    int i,j;
    while(1)
    {
        scanf("%d%d",&n,&k);
        if(n==-1&&k==-1)
```

```
                break;
            memset(lie,0,sizeof(lie));
            for(i=0;i<n;i++)
                scanf("%s",s[i]);
            c=0;
            for(i=0;i<=n-k;i++)       //dfs(i,k)表示从第 i 行开始向下放 k 个棋子，因为一行只能放一个，所以最低从 n-k 行开始放
                dfs(i,k);
            printf("%d\n",c);
        }
        return 0;
}
```

102.帮助吉米

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct a
{
        int zuo;
        int you;
        int h;
}lou[1001];
int dp[1001][3];
int n,x,y,max;
int min(int c,int d)
{
        if(c>=d)        return d;
        else            return c;
}
int comp(const void *c,const void *d)
{
        return ((a *)c)->h-((a *)d)->h>0?1:-1;
}
int main(void)
{
        //freopen("in.txt","r",stdin);
        int i,j,k,t,w,l;
        scanf("%d",&t);
        while(t--)
        {
                w=0;
                scanf("%d%d%d%d",&n,&x,&y,&max);
                for(i=0;i<=n;i++)
                        memset(dp[i],0,sizeof(dp[i]));
                for(i=0;i<n;i++)
                        scanf("%d%d%d",&lou[i].zuo,&lou[i].you,&lou[i].h);
                lou[n].zuo=x;
                lou[n].you=x;
                lou[n].h=y;
                qsort(lou,n+1,sizeof(lou[0]),comp);
                dp[0][1]=0;
                dp[0][2]=0;
                for(i=1;i<=n;i++)                       //从第 1 层一直到第 n-1 层
                {
                        for(j=i-1;j>=0;j--)             //每一层找他下面最近的可以着陆一层
                        {
                                if(lou[i].h>lou[j].h&&lou[i].zuo>=lou[j].zuo&&lou[i].zuo<=lou[j].you)
                                {
```

```
                        if(lou[i].h-lou[j].h<=max)
                        {

    dp[i][1]=min(lou[i].zuo-lou[j].zuo+dp[j][1],lou[j].you-lou[i].zuo+dp[j][2]);
                            break;
                        }
                        else
                        {
                            dp[i][1]=99999999;
                            break;
                        }
                    }
                }
                if(j==-1&&lou[i].h<=max)
                    dp[i][1]=0;
                else if(j==-1&&lou[i].h>max)
                    dp[i][1]=99999999;
                for(k=i-1;k>=0;k--)
                {
                    if(lou[i].h>lou[k].h&&lou[i].you>=lou[k].zuo&&lou[i].you<=lou[k].you)
                    {
                        if(lou[i].h-lou[k].h<=max)
                        {

    dp[i][2]=min(lou[i].you-lou[k].zuo+dp[k][1],lou[k].you-lou[i].you+dp[k][2]);
                            break;
                        }
                        else
                        {
                            dp[i][2]=99999999;
                            break;
                        }
                    }
                }
                if(k==-1&&lou[i].h<=max)
                    dp[i][2]=0;
                else if(k==-1&&lou[i].h>max)
                    dp[i][2]=99999999;
            }
            printf("%d\n",min(dp[n][1],dp[n][2])+y);
    }
    return 0;
}
```

103.最长公共子序列

```
#include<stdio.h>
#include<string.h>
char s1[1000];
char s2[1000];
int dp[1000][1000];
int max(int a,int b)
{
    return (a>b)?a:b;
}
int main(void)
{
    int i,j,k;
    while(scanf("%s%s",s1,s2)==2)
    {
```

```
            for(i=0;i<1000;i++)
                memset(dp[i],0,sizeof(dp[i]));
            if(s1[0]==s2[0])
                dp[0][0]=1;
            else
                dp[0][0]=0;
            for(i=0;i<strlen(s1);i++)
            {
                for(j=0;j<strlen(s2);j++)
                {
                    if(s1[i]==s2[j]&&i==0||s1[i]==s2[j]&&j==0)
                        dp[i][j]=1;
                    else if(s1[i]==s2[j])
                        dp[i][j]=dp[i-1][j-1]+1;
                    else if(s1[i]!=s2[j]&&i==0&&j!=0)
                        dp[i][j]=dp[i][j-1];
                    else if(s1[i]!=s2[j]&&j==0&&i!=0)
                        dp[i][j]=dp[i-1][j];
                    else if(s1[i]!=s2[j]&&i!=0&&j!=0)
                        dp[i][j]=max(dp[i][j-1],dp[i-1][j]);
                }
            }
            printf("%d\n",dp[strlen(s1)-1][strlen(s2)-1]);
        }
        return 0;
}
```

104.陪审团的人选（copy）

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

int compare(void const* a, void const* b)
{
    return *(int*)a - *(int*)b;
}

int main()
{
    int n, m, i, j, k, t1, t2, casenum=1, min;
    int f[21][1000], p[201], d[201], path[21][1000];
    int Qnum[21], Q[21][900]; //队列
    int ans[21];

    while (scanf("%d %d", &n, &m) && n)
    {
        memset(f, -1, sizeof(f));    //这样也行！！赋值之后是-1;为什么呢？？？
        memset(Qnum, 0, sizeof(Qnum));
        for (i=1; i<=n; i++)
        {
            scanf("%d %d", &p[i], &d[i]);
        }

        Q[0][0] = 400;
        Qnum[0] = 1;
        f[0][400] = 0;
        for (i=0; i<m; i++)
        {
```

```
                for (j=0; j<Qnum[i]; j++)
                {
                    for (k=1; k<=n; k++)
                    {
                        if(f[i][Q[i][j]]+p[k]+d[k] > f[i+1][Q[i][j]+p[k]-d[k]])
                        {
                            t1 = i;             //以下检验 k 是否已经在 f[i][Q[i][j]]中被选上
                            t2 = Q[i][j];
                            while (t1>0 && path[t1][t2]!=k)
                            {
                                t2 -= p[path[t1][t2]]-d[path[t1][t2]];
                                t1--;
                            }
                            if (t1==0) //没有被选上，可选
                            {
                                if (f[i+1][Q[i][j]+p[k]-d[k]] == -1)
                                {       //添加到队列
                                    Q[i+1][Qnum[i+1]] = Q[i][j]+p[k]-d[k];
                                    Qnum[i+1]++;
                                }
                                f[i+1][Q[i][j]+p[k]-d[k]] = f[i][Q[i][j]]+p[k]+d[k];
                                path[i+1][Q[i][j]+p[k]-d[k]] = k;
                            }
                        }
                    }
                }
            }
            min = 900;          //计算控方和辩方差值的绝对值最小值
            for (i=0; i<Qnum[m]; i++)
            {
                if (abs(Q[m][i]-400) < min) min = abs(Q[m][i]-400);
            }
            if (f[m][400+min] > f[m][400-min]) min = min+400;   //选最小差值中总分和最大的
            else min = 400-min;
            printf("Jury #%d\n", casenum++);
            printf("Best jury has value %d for prosecution and value %d for defence:\n",
                                        (min-400+f[m][min])/2, (400-min+f[m][min])/2);

            for (i=0; i<m; i++) //从后往前依次找被选中的 m 个人
            {
                ans[i] = path[m-i][min];
                min -= p[ans[i]]-d[ans[i]];
            }
            qsort(ans, m, sizeof(int), compare);

            for (i=0; i<m; i++)
               printf(" %d", ans[i]);
            printf("\n\n");
        }
}
```

105.矩形覆盖

```
#include<iostream>
#include<stdio.h>
#include<cmath>
using namespace std;
#define INF 99999999
int n,m;
int x[16],y[16];
```

```
int dp[1<<15];
int s[16*16];
int state[16*16];
void gets(int i,int j)
{
int k=(1<<i)+(1<<j);
for(int t=0;t<n;t++)
if((x[i]-x[t])*(x[t]-x[j])>=0&&(y[i]-y[t])*(y[t]-y[j])>=0)
k|=(1<<t);
int are=0;
if(x[i]==x[j])are=abs(y[i]-y[j]);
else if(y[i]==y[j])are=abs(x[i]-x[j]);
else are=abs(x[i]-x[j])*abs(y[i]-y[j]);
state[m]=k;
s[m++]=are;
return;
}
int main()
{
int i,j;
while(scanf("%d",&n)==1&&n!=0)
{
for(i=0;i<n;i++)
scanf("%d%d",&x[i],&y[i]);
m=0;
for(i=1;i<n;i++)
for(j=0;j<i;j++)
gets(i,j);
for(i=1;i<(1<<n);i++)
dp[i]=INF;
dp[0]=0;
for(i=0;i<(1<<n);i++)
for(j=0;j<m;j++)
{
int temp=i|state[j];
if(temp==i)continue;
dp[temp]=min(dp[temp],dp[i]+s[j]);
}
printf("%d\n",dp[(1<<n)-1]);
}
}
```

106.金银岛

```
#include<stdio.h>
#include<stdlib.h>
struct wu
{
	int zhong;
	int jiazhi;
	double bi;
}wupin[101];
int comp(const void *a,const void *b)
{
	return ((wu *)a)->bi<((wu *)b)->bi?1:-1;
}
int main(void)
{
	int i,j,k,t,m,n;
	scanf("%d",&t);
```

```
	while(t--)
	{
		double sum=0;
		scanf("%d",&m);
		scanf("%d",&n);
		for(i=0;i<n;i++)
			scanf("%d%d",&wupin[i].zhong,&wupin[i].jiazhi);
		for(i=0;i<n;i++)
			wupin[i].bi=(double)wupin[i].jiazhi/(double)wupin[i].zhong;
		qsort(wupin,n,sizeof(wupin[0]),comp);
		for(i=0;i<n;i++)
		{
			if(m>=wupin[i].zhong)
			{
				sum+=(double)wupin[i].jiazhi;
				m-=wupin[i].zhong;
			}
			else if(m>0&&m<wupin[i].zhong)
			{
				sum+=double(m)/(double)wupin[i].zhong*(double)wupin[i].jiazhi;
				break;
			}
			else
				break;
		}
		printf("%.2lf\n",sum);

	}
	return 0;
}
```

107. 1097 Pell 数列

```
#include<stdio.h>
long long dp[1000001];
int main(void)
{
	int i,j,k,n,m;
	dp[1]=1;
	dp[2]=2;
	for(i=3;i<=1000000;i++)
	{
		dp[i]=2*dp[i-1]+dp[i-2];
		while(dp[i]>=32767)
			dp[i]-=32767;
	}
	scanf("%d",&n);
	while(n--)
	{
		scanf("%d",&k);
		printf("%d\n",dp[k]%32767);
	}
	return 0;
}
```

108.木材分割

```
#include<stdio.h>
#include<stdlib.h>
int comp(const void *a,const void *b)
{
	return *(int *)a-*(int *)b;
```

```
}
int mu[10001];
int cc[10001];
int main(void)
{
	int i,j,k,n,l,sum=0;
	scanf("%d%d",&n,&k);
	for(i=0;i<n;i++)
	{
		scanf("%d",&mu[i]);
		sum+=mu[i];
	}
	if(sum<k)
	{
		printf("0");
	}
	else
	{
		qsort(mu,n,sizeof(mu[0]),comp);
		for(i=mu[n-1];;i--)
		{
			for(l=0;l<n;l++)
				cc[l]=mu[l];
			sum=0;
			for(j=0;j<n;j++)
			{
				while(cc[j]>=i)
				{
					cc[j]-=i;
					sum++;
				}
			}
			if(sum>=k)
			{
				printf("%d",i);
				break;
			}
		}
	}
	return 0;
}
```

109.All in all

```
#include<stdio.h>
#include<string.h>
char s1[100000];
char s2[100000];
int main(void)
{
	long i,j,k;
	while(scanf("%s%s",s1,s2)==2)
	{
		long len1,len2;
		len1=strlen(s1);
		len2=strlen(s2);
		i=0;j=0;
		while(1)
		{
			if(i==len1)
```

```
                {
                        printf("Yes\n");
                        break;
                }
                else if(j==len2&&i<len1)
                {
                        printf("No\n");
                        break;
                }
                else if(s1[i]==s2[j])
                {
                        i++;
                        j++;
                }
                else
                        j++;
        }
    }
    return 0;
}
```

110.五子棋判输赢

```
#include<stdio.h>
int dp[1000][1000];
int n;
int f(int x,int y,int s)
{
    int i,j,k,l,ok;
    for(i=y-4;i<=y;i++)
    {
        if(i>=0&&i+4<n)
        {
            ok=1;
            for(j=i;j<=i+4;j++)
                if(dp[x][j]!=s)
                {
                    ok=0;
                    break;
                }
            if(ok==1)
                return 1;
        }
    }
    for(i=x-4;i<=x;i++)
    {
        if(i>=0&&i+4<n)
        {
            ok=1;
            for(j=i;j<=i+4;j++)
                if(dp[j][y]!=s)
                {
                    ok=0;
                    break;
                }
            if(ok==1)
                return 1;
        }
    }
    for(i=x-4,j=y-4;i<=x&&j<=y;i++,j++)
```

```
	{
		if(i>=0&&i+4<n&&j>=0&&j+4<n)
		{
			ok=1;
			for(k=i,l=j;k<=i+4&&l<=j+4;k++,l++)
				if(dp[k][l]!=s)
				{
					ok=0;
					break;
				}
			if(ok==1)
				return 1;
		}
	}
	for(i=x-4,j=y+4;i<=x&&j>=y;i++,j--)
	{
		if(i>=0&&i+4<n&&j-4>=0&&j<n)
		{
			ok=1;
			for(k=i,l=j;k<=i+4&&l>=j-4;k++,l--)
				if(dp[k][l]!=s)
				{
					ok=0;
					break;
				}
			if(ok==1)
				return 1;
		}
	}
	return 0;
}
int main(void)
{
	int i,j,k,m,x,y;
	scanf("%d%d",&n,&m);
	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			dp[i][j]=-1;
	for(i=1;i<=m;i++)
	{
		scanf("%d%d",&x,&y);
		dp[x][y]=0;
		if(f(x,y,0)==1)
		{
			printf("%d %d",i,0);
			break;
		}
		i++;
		scanf("%d%d",&x,&y);
		dp[x][y]=1;
		if(f(x,y,1)==1)
		{
			printf("%d %d",i,1);
			break;
		}
	}
	return 0;
}
```

111.最大子序列和

```
#include<stdio.h>
int left[100001];
int right[100001];
int num[100001];
int main(void)
{
	int i,j,k,n,max1,sum;
	while(scanf("%d",&n)==1)
	{
		if(n==0)
			break;
		for(i=1;i<=n;i++)
			scanf("%d",&num[i]);
		max1=-99999999;
		sum=0;
		for(i=1;i<=n;i++)
		{
			sum=sum+num[i];
			if(sum>max1)
				max1=sum;
			if(sum<0)
				sum=0;
			left[i]=max1;
		}
		max1=-99999999;
		sum=0;
		for(i=n;i>=1;i--)
		{
			sum=sum+num[i];
			if(sum>max1)
				max1=sum;
			if(sum<0)
				sum=0;
			right[i]=max1;
		}
		max1=-9999999;
		for(i=1;i<n;i++)
		{
			if(left[i]+right[i+1]>max1)
				max1=left[i]+right[i+1];
		}
		printf("%d\n",max1);
	}
	return 0;
}
```

112.炮兵阵地

```
#include <stdio.h>
#include <string.h>
#include <iostream>
#define MAX(a,b) (a)>(b)?(a):(b)
using namespace std;
int dp[105][65][65];	//d[i][j][k]: “第 i 行状态是 s[j]，第 i-1 行状态是 s[k]”的
int s[105];	//一行的状态选择 s[0], s[1], ... , s[k-1]
int n,m;	//n 行×m 列
int k;	//一行的所有状态数
int map[105];	//'H"P'地图 map[0]~map[n-1]，地图每一行 map[line]: 1001 表示 HPPH
int sum[105];
```

```
/*
    很久就看推荐题目有这个了，一直没做，因为看了好几次没看懂，都说 dp，这几天看了状态压缩后明白了，其实就是用
    二进制来表示各个位置的状态然后进行枚举，把状态放进数组里就行，在这里用 dp[i][j][k]表示第 i 行，当前 j 状态，
    i-1 行是 k 状态时候的最大炮数  dp[i][j][k]=MAX(dp[i][j][k],dp[i-1][k][p]+sum[j])

    CAUTION:
    1. 所有下标均从 0 开始
    2. m<=10 保证了可以用一个 int 存储一行的状态
*/

//状态 s[x]是否造成行冲突
bool ok(int x)
{
    if(x&(x<<1))return false;
    if(x&(x<<2))return false;
    return true;
}

//状态 s[x]下有多少个 1
int getsum(int x)
{
    int num=0;
    while(x>0)
    {
        if(x&1)num++;
        x>>=1;
    }
    return num;
}

void find()
{
    memset(s,0,sizeof(s));
    for(int i=0;i<(1<<m);i++) //i 枚举所有 m 位的二进制数
    {
        if(ok(i))
        {
            s[k]=i;
            sum[k++]=getsum(i);
        }

    }
}

int main()
{
    while(scanf("%d%d",&n,&m)!=EOF){
        memset(dp,-1,sizeof(dp));

        int i;
        for(i=0;i<n;i++){
            for(int j=0;j<m;j++){
                char tmp;
                cin>>tmp;
                if(tmp=='H')map[i]=map[i]|(1<<j);//把第 i 行原始状态取反后放入 map[i]
```

```
            }
        }

        k=0;
        find();

        //1. 初始化第 0 行状态（只考虑有效状态，无效状态为-1）
        for(i=0;i<k;i++)
            if(!(s[i]&map[0])) //s[i]为 1 的位如果对应平原(0)，则&运算后为 0
                dp[0][i][0]=sum[i];

        //2. 计算第 1~n-1 行状态（碰到无效状态,continue）
        for(int r=1;r<n;r++)
        {
            for(int i=0;i<k;i++)//枚举第 r 行的状态  s[i]
            {
                if(map[r]&s[i])       continue; //通过地形排除部分第 r 行的状态

                for(int p=0;p<k;p++)     //枚举第 r-1 行状态  s[p]
                {
                    if(s[i] & s[p])  continue; //r 与 r-1 没有想接触的

                    for(int q=0;q<k;q++)     //枚举第 r-2 行状态 s[q]
                    {
                        if(s[p] & s[q])      continue;     //Sam：这行是我加的
                        if(s[i] & s[q])continue; //r 与 r-2 行没有接触的

                        if(dp[r-1][p][q]==-1)    continue; //所有不可能的情形 dp[i][j][k]
都为-1（初始化的值）
                        dp[r][i][p]=MAX(dp[r][i][p],dp[r-1][p][q]+sum[i]);
                    }
                }
            }
        }

        int ans=0;
        for(i=0;i<k;i++)
            for(int j=0;j<k;j++)
                ans=MAX(ans,dp[n-1][i][j]);
        printf("%d\n",ans);
    }
    return 0;
}
113. 碱基的忧伤--计算机系男生的故事之一
#include<stdio.h>
#include<string.h>
int dp[10000][10000];   //dp[i][j]表示第一排前 i 个碱基和第二排前 j 个碱基组合的最大匹配值
char s1[10000];
char s2[10000];
int map[4][4];        //记录 AGTC 对应分数
int score(char x,char y)
{
    if(x=='A'&&y=='A')
        return map[0][0];
    if(x=='A'&&y=='G'||y=='A'&&x=='G')
        return map[0][1];
    if(x=='A'&&y=='C'||y=='A'&&x=='C')
        return map[0][2];
```

```
    if(x=='A'&&y=='T'||y=='A'&&x=='T')
        return map[0][3];
    if(x=='G'&&y=='G')
        return map[1][1];
    if(x=='G'&&y=='C'||y=='G'&&x=='C')
        return map[1][2];
    if(x=='G'&&y=='T'||y=='G'&&x=='T')
        return map[1][3];
    if(x=='C'&&y=='C')
        return map[2][2];
    if(x=='C'&&y=='T'||y=='C'&&x=='T')
        return map[2][3];
    if(x=='T'&&y=='T'||y=='T'&&x=='T')
        return map[3][3];
}
int max(int a,int b,int c)
{
    if(a>=b)
    {
        if(c>=a)
            return c;
        else
            return a;
    }
    else
    {
        if(c>=b)
            return c;
        else
            return b;
    }
}
int main(void)
{
    //freopen("in.txt","r",stdin);
    int i,j,k,l,t,d,sum,n,w1,w2;
    int m;          //有多少组
    scanf("%d",&m);
    for(t=1;t<=m;t++)
    {
        for(i=0;i<=3;i++)
            for(j=0;j<=3;j++)
                scanf("%d",&map[i][j]);
        scanf("%d",&d);                 //占位符惩罚权重 d 的值
        scanf("%d",&n);                 //需要比对的 DNA 片段的对数 N
        printf("Case #%d\n",t);
        while(n--)
        {
            scanf("%s",s1);
            scanf("%s",s2);
            w1=strlen(s1);
            w2=strlen(s2);
            dp[0][0]=0;
            for(i=1;i<=w1;i++)
                dp[i][0]=i*d;
            for(i=1;i<=w2;i++)
                dp[0][i]=i*d;
            for(i=1;i<=w1;i++)                          //开始 dp
```

```
            {
                for(j=1;j<=w2;j++)
                {
dp[i][j]=max(dp[i-1][j]+d,dp[i-1][j-1]+score(s1[i-1],s2[j-1]),dp[i][j-1]+d);
                }
            }
            printf("%d\n",dp[w1][w2]);
        }
    }
    return 0;
}
```

114.扩号匹配问题

```
#include<stdio.h>
#include<string.h>
char s[1000];
int dp[1000];
int main(void)
{
    //freopen("in.txt","r",stdin);
    int i,j,k,n;
    scanf("%d",&n);
    while(scanf("%s",s)==1)
    {
        memset(dp,0,sizeof(dp));
        n=strlen(s);
        for(i=0;i<n;i++)
        {
            if(s[i]=='(')
                dp[i]=-1;
            if(s[i]==')')
                dp[i]=1;
        }
        for(i=0;i<n;i++)
        {
            if(s[i]==')')
            {
                for(j=i-1;j>=0;j--)
                {
                    if(s[j]=='('&&dp[j]==-1)
                    {
                        dp[i]=0;
                        dp[j]=0;
                        break;
                    }
                }
            }
        }
        printf("%s\n",s);
        for(i=0;i<n;i++)
        {
            if(dp[i]==-1)
                printf("$");
            else if(dp[i]==1)
                printf("?");
            else
            printf(" ");
        }
```

```
            printf("\n");
        }
        return 0;
}115. 最多点数直线问题
#include<iostream>
using namespace std;
#define MAX 705
struct Point{
        int x,y;
}p[MAX];
int main()
{
        int n,i,j,k,a,b;
        while(cin>>n&&n)
        {
                for(i=0;i<n;i++)
                        cin>>p[i].x>>p[i].y;
                int sum,max=0;
                for(i=0;i<n;i++)
                        for(j=i+1;j<n;j++)
                        {
                                sum=2;
                                for(k=j+1;k<n;k++)
                                {
                                        a=(p[i].y-p[k].y)*(p[j].x-p[k].x);
                                        b=(p[i].x-p[k].x)*(p[j].y-p[k].y);
                                        if(a==b) sum++;
                                }
                                if(max<sum) max=sum;
                        }
                        cout<<max<<endl;
        }
        return 0;
}
116.最简区间
#include<iostream>
#include<algorithm>
#include<cstdio>
using namespace std;
struct node
{
 int a;
 int b;
 bool operator <(const node & other)const
 {
  if(a<other.a) return true;
  else if(a==other.a)
  {
   if(b<other.b)return true;
   else return false;
  }
  else
   return false;
 }

}num[50010];

int main()
```

```
{
	//freopen("in.txt","r",stdin);
 int n,a,b;
 scanf("%d",&n);
 for(int i=0;i<n;i++)
 {
  scanf("%d%d",&num[i].a,&num[i].b);
 }

 sort(num,num+n);
 a=num[0].a;
 b=num[0].b;
 for(int i=1;i<n;i++)
 {
  if(b<num[i].a)
  {
   printf("%d %d\n",a,b);
   a=num[i].a;
   b=num[i].b;
  }

  else if(b<num[i].b)
   b=num[i].b;
 }
 printf("%d %d\n",a,b);
 return 0;
}
```

117.三值排序

```
#include <stdio.h>
int a[1001];
int compare(int s1,int e1,int num1,int s2,int e2,int num2){
	int i,j,sum=0;
	for(i=s1,j=s2;i<s1+e1 && j<s2+e2;i++,j++){
		if(a[i]==num2 || a[j]==num1){
			if(a[j]!=num1){
				for(;j<s2+e2;j++)
					if(a[j]==num1)
						break;
			}
			else if(a[i]!=num2){
				for(;i<s1+e1;i++)
					if(a[i]==num2)
						break;
			}
			if(i==s1+e1 || j==s2+e2)
				break;
			a[i]^=a[j];
			a[j]^=a[i];
			a[i]^=a[j];
			sum++;
		}
	}
	return sum;
}

void solve(int n){
	int i,sum=0,count=0;
	int num[3]={0};
```

```
        for(i=0;i<n;i++)
                num[a[i]-1]++;
        sum+=compare(0,num[0],1,num[0],num[1],2);
        sum+=compare(num[0],num[1],2,num[0]+num[1],num[2],3);
        sum+=compare(0,num[0],1,num[0]+num[1],num[2],3);
        for(i=0;i<num[0];i++){
                if(a[i]!=1)
                        count++;
        }
  for(;i<num[0]+num[1];i++){
    if(a[i]!=2)
      count++;
  }
  for(;i<n;i++){
    if(a[i]!=3)
      count++;
  }
        sum+=count/3*2;
        printf("%d\n",sum);
}
int main(){
        //freopen("in.txt","r",stdin);
        int n;
        while(scanf("%d",&n)!=EOF){
                int i;
                for(i=0;i<n;i++)
                        scanf("%d",&a[i]);
                solve(n);
        }
        return 0;
}
```

118. （附加题）公交线路

```
#include<cmath>
#include<iostream>
#include<cstdio>
#include<algorithm>
#include <vector>
using namespace std;
class Route
{
public:
        Route(int firstTime_, int interval_, int frequence_)
                : firstTime(firstTime_)
                , interval(interval_)
                , frequence(frequence_)
        {
        }
        bool operator <(const Route &lhs) const
        {
                return frequence > lhs.frequence;
        }
        int firstTime;
        int interval;
        int frequence;
};
int visited[60] = { 0 };
std::vector <Route> routes;
void prepareRoutes()
```

```
{
	routes.reserve(900);
	for (int firstTime = 0; firstTime <= 29; ++firstTime)
	{
		if (!visited[firstTime])
			continue;
		for (int interval = firstTime + 1; interval <= 59 - firstTime; ++interval)
		{
			bool suffice = true;
			for (int k = firstTime + interval; k <= 59; k += interval)
			{
				if (!visited[k])
				{
					suffice = false;
					break;
				}
			}
			if (suffice)
			{
				int fre = 1 + (59 - firstTime) / interval;
				routes.push_back(Route(firstTime, interval, fre));
			}
		}
	}
	sort(routes.begin(), routes.end());
}//prepareRoutes

int nMinRoutes = 17;
bool isValidRoute(int firstTime, int interval)
{
	int t = firstTime;
	for (; t <= 59; t += interval)
	{
		if (!visited[t])
			return false;
	}
	return true;
}
void solve(int currentRouteNumber, int nCurrentRoutes, int nRemain)
{
	if (nRemain == 0)
	{
		if (nCurrentRoutes < nMinRoutes)
		{
			nMinRoutes = nCurrentRoutes;
			//bestX = currentX;
		}
		//else
		//{
		//	currentX.clear();
		//}
		return;
	}
	while (currentRouteNumber < (int)routes.size() &&
routes[currentRouteNumber].frequence > nRemain)
		++currentRouteNumber;
	for (;currentRouteNumber < (int)routes.size(); ++currentRouteNumber)
	{
```

```
        if (nCurrentRoutes + 1 + (nRemain - 1) / routes[currentRouteNumber].frequence >=
nMinRoutes)
            return;

        if (isValidRoute(routes[currentRouteNumber].firstTime,
routes[currentRouteNumber].interval))
        {
            //currentX.push_back(routes[currentRouteNumber]);

            for (int i = routes[currentRouteNumber].firstTime; i <= 59; i +=
routes[currentRouteNumber].interval)
            {
                --visited[i];
                --nRemain;
            }
            solve(currentRouteNumber, nCurrentRoutes + 1, nRemain);//线路不同的公交路
线可能有相同的起始时间和间隔
            for (int i = routes[currentRouteNumber].firstTime; i <= 59; i +=
routes[currentRouteNumber].interval)
            {
                ++visited[i];
                ++nRemain;
            }
            //currentX.pop_back();
        }
    }
}
int main()
{
    int n;
    scanf ("%d", &n);
    int tmp;
    for (int i = 0; i < n; ++i)
    {
        scanf("%d", &tmp);
        visited[tmp]++;
    }
    prepareRoutes();
    solve(0, 0, n);
    printf("%d\n", nMinRoutes);
    return 0;
}
```

119.填充箱子

```
#include<iostream>
using namespace std;
int max(int a,int b)
{
    return a>b?a:b;
}
int main(void)
{
    int s1,s2,s3,s4,s5,s6;          //6 种 size 的盒子数量
    while(cin>>s1>>s2>>s3>>s4>>s5>>s6 && (s1+s2+s3+s4+s5+s6))
    {
        int BoxNum=0;                   //放进所有盒子所需的最少箱子数

        BoxNum+=s6;                     //6*6 的盒子，每个都刚好独占一个箱子
```

```
        BoxNum+=s5;                 //5*5 的盒子，放进箱子后，每个箱子余下的空间只能放 11 个 1*1 的盒子
        s1=max(0,s1-s5*11);       //把 1*1 的盒子尽可能地放进已放有一个 5*5 盒子的箱子

        BoxNum+=s4;                 //4*4 的盒子，放进箱子后，每个箱子余下的空间为 5 个 2*2 的盒子空间
                                    //先把所有 2*2 的盒子尽可能地放进这些空间
        if(s2>=s4*5)                //若 2*2 的盒子数比空间多
            s2-=s4*5;               //则消去已放进空间的部分
        else                        //若 2*2 的盒子数比空间少
        {                           //则先把所有 2*2 的盒子放进这些空间
            s1=max(0,s1-4*(s4*5-s2));    //再用 1*1 的盒子填充本应放 2*2 盒子的空间
            s2=0;                   //一个 2*2 空间可放 4 个 1*1 盒子
        }

        BoxNum+=(s3+3)/4;           //每 4 个 3*3 的盒子完全独占一个箱子
        s3%=4;                  //3*3 的盒子不足 4 个时，都放入一个箱子，剩余空间先放 2*2，再放 1*1
        if(s3)
        {                           //当箱子放了 i 个 3*3 盒子，剩下的空间最多放 j 个 2*2 盒子
            if(s2>=7-2*s3)          //其中 i={1,2,3} ; j={5,3,1}  由此可得到条件的关系式
            {
                s2-=7-2*s3;
                s1=max(0,s1-(8-s3));  //当箱子放了 i 个 3*3 盒子，并尽可能多地放了个 2*2 盒子后
            }                           //剩下的空间最多放 j 个 1*1 盒子，其中 i={1,2,3} ; j={7,6,5}
            else                    //但当 2*2 的盒子数不足时，尽可能把 1*1 盒子放入剩余空间
            {   //一个箱子最多放 36 个 1*1，一个 3*3 盒子空间最多放 9 个 1*1，一个 2*2 盒子空间最多放 4 个 1*1
                s1=max(0,s1-(36-9*s3-4*s2));      //由此很容易推出剩余空间能放多少个 1*1
                s2=0;
            }
        }

        BoxNum+=(s2+8)/9;           //每 9 个 2*2 的盒子完全独占一个箱子
        s2%=9;                  //2*2 的盒子不足 9 个时，都放入一个箱子，剩余空间全放 1*1
        if(s2)
            s1=max(0,s1-(36-4*s2));

        BoxNum+=(s1+35)/36;       //每 36 个 1*1 的盒子完全独占一个箱子

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

120.POJ1328 雷达装置

```
#include<iostream>
#include<math.h>
using namespace std;
const int island_max=1000;
int main(void)
{
    int i,j;
    double x[island_max],y[island_max];
```

```
    double num,rad;
    for(;;)
    {
        /*Input end*/
        cin>>num>>rad;
        if(!(num&&rad))break;
        static int count=1;   //记录 case 数目
        /*read in coordinate*/
        bool flag=false;
        for(i=0;i<num;i++)
        {
            cin>>x[i]>>y[i];
            if(y[i]>rad)
            flag=true;
        }
        /*case fail*/
        if(flag)
        {
            cout<<"Case "<<count++<<": -1"<<endl;
            continue;
        }

        /*bubble sort*/
        //这里由于 y 要随 x 连带排序，不能简单地使用 快排 qsort
        double temp;
        for(i=0;i<num-1;i++)
            for(j=0;j<num-i-1;j++)
                if(x[j]>x[j+1])
                {
                    temp=x[j];
                    x[j]=x[j+1];
                    x[j+1]=temp;
                    temp=y[j];
                    y[j]=y[j+1];
                    y[j+1]=temp;
                }
        double left[island_max],righ[island_max];   //海岛圆在海岸线上的左右交点
        for(i=0;i<num;i++)
        {
            left[i]=x[i]-sqrt(rad*rad-y[i]*y[i]);
            righ[i]=x[i]+sqrt(rad*rad-y[i]*y[i]);
        }
        int radar=1;
        for(i=0,temp=righ[0];i<num-1;i++)
            if(left[i+1]>temp)
            {
                temp=righ[i+1];
                radar++;
            }
            else if(righ[i+1]<temp)
                temp=righ[i+1];
        cout<<"Case "<<count++<<": "<<radar<<endl;
    }
    return 0;
}
```

121.POJ1019 数字序列

```
#include<iostream>
#include<math.h>
```

```
using namespace std;
const int size=31269;
unsigned a[size];
unsigned s[size];
void play_table(void)
{
	a[1]=s[1]=1;
	for(int i=2;i<size;i++)
	{
		a[i]=a[i-1]+(int)log10((double)i)+1;
		s[i]=s[i-1]+a[i];
	}
	return;
}
int compute(int n)
{
	int i=1;
	while(s[i]<n)
		i++;

	int pos=n-s[i-1];

	int len=0;
	for(i=1;len<pos;i++)
		len+=(int)log10((double)i)+1;

	return (i-1)/(int)pow((double)10,len-pos)%10;
}
int main(void)
{
	play_table();
	int test;
	cin>>test;
	while(test--)
	{
		int n;
		cin>>n;
		cout<<compute(n)<<endl;
	}
	return 0;
}
```

122.POJ1154 字母

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string.h>
using namespace std;
int visit[30];
int n,m;
int step[8] = {1,0,-1,0,0,1,0,-1};
int num[100][100],cou = 1,mmax = 0;;
void DFS(int i, int j)
{
	for( int k=0; k<8; k+=2)
	{
		int a = i+step[k];
		int b = j+step[k+1];
		if( a>=1 && a<=n && b>=1 && b<=m && !visit[num[a][b]] )
```

```
            {
                visit[num[a][b]] = 1;
                cou++;
                DFS(a,b);
                if( cou > mmax )
                    mmax = cou;
                cou--;
                visit[num[a][b]] = 0;
            }
        }
}
int main(void)
{
    char str[100];
    while( cin >> n >> m )
    {
        getchar();
        memset(visit,0,sizeof(visit));
        cou = 1; mmax = 1;
        for(int i=1; i<=n; i++)
        {
            gets(str);
            for(int k=0; k<m; k++)
                num[i][k+1] = str[k] -'A' + 1;
        }
        visit[num[1][1]] = 1;
        DFS(1,1);
        cout << mmax << endl;
    }
return 0;
}
```

123.集合选数

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<cmath>
#include<algorithm>
#define MOD 1000000001
#define MAXN 20
#define MAXM 15
#define N 100010
using namespace std;
typedef long long LL;
int n,r,c,map[MAXN][MAXM];
LL ans=1,F[2][1<<12];
bool use[N];
int get(int k,int p){    return k>>(p-1)&1;    }
void MakeMatrix(int p)
{
    memset(map,0,sizeof(map));
    r=c=1;
    for(int i=1,d1=p;d1<=n;i++,d1*=2)
    {
        use[d1]=true,map[i][1]=d1,r=max(r,i);
        for(int j=2,d2=d1*3;d2<=n;j++,d2*=3)
            use[d2]=true,map[i][j]=d2,c=max(c,j);
    }
}
```

```
LL Dp()
{
	LL ans=0;
	int mark=0,M=(1<<c)-1;
	for(int i=1;i<=M;i++)
		F[mark][i]=0;
	F[mark][0]=1;
	for(int i=1;i<=r;i++)
		for(int j=1;j<=c;j++)
		{
			for(int k=0;k<=M;k++)
				F[mark^1][k]=0;
			for(int k=0;k<=M;k++)
			{
				if(F[mark][k]==0)	continue;
				F[mark^1][k<<1&M]=(F[mark^1][k<<1&M]+F[mark][k])%MOD;
				if(map[i][j]&&j==1&&!get(k,c))
F[mark^1][k<<1|1&M]=(F[mark^1][k<<1|1&M]+F[mark][k])%MOD;
				if(map[i][j]&&j!=1&&!get(k,c)&&!get(k,1))
F[mark^1][k<<1|1&M]=(F[mark^1][k<<1|1&M]+F[mark][k])%MOD;
			}
			mark^=1;
		}
	for(int i=0;i<=M;i++)
		ans=(ans+F[mark][i])%MOD;
	return ans;
}
void work()
{
	for(int i=1;i<=n;i++)
		if(use[i]==false)
		{
			MakeMatrix(i);
			ans=ans*Dp()%MOD;
		}
}
int main()
{
	scanf("%d",&n);
	work();
	printf("%lld\n",ans);
	return 0;
}
```

124.判断是否为 C 语言的合法标识符

```
#include<stdio.h>
#include<string.h>
char s[100];
int main(void)
{
	int i,j,k;
	char c;
	int n;
	scanf("%d",&n);
	while(n--)
	{
		while(1)
		{
			c=getchar();
```

```
            if(c!='\n')
                break;
        }
        i=0;
        while(c!='\n')
        {
            s[i++]=c;
            c=getchar();
        }
        s[i]='\0';
        //printf("%s\n",s);
        if(s[0]=='_'||s[0]>='a'&&s[0]<='z'||s[0]>='A'&&s[0]<='Z')
        {
            int ok=1;
            for(i=1;i<strlen(s);i++)
            {
                if(s[i]=='_'||s[i]>='a'&&s[i]<='z'||s[i]>='A'&&s[i]<='Z'||s[i]>='0'&&s[i]<='9')
                    ;
                else
                {
                    ok=0;
                    break;
                }
            }
            if(ok==1)
                printf("1\n");
            else
                printf("0\n");
        }
        else
            printf("0\n");
    }
    return 0;
}
```

125.称体重

```
#include<iostream>
using namespace std;
void sort(int a,int b,int c,int d)
{
    int result[4]={a,b,c,d};
    char name[4]={'z','q','s','l'};
    int temp;
    char tempname;
    for(int i=0;i<4;i++)
    {
        for(int j=i;j<4;j++)
        {
            if(result[i]<result[j])
            {
                temp=result[i];
                result[i]=result[j];
                result[j]=temp;
                tempname=name[i];
                name[i]=name[j];
                name[j]=tempname;
            }
        }
    }
```

```
    for(int i=0;i<4;i++)
    {
        cout<<name[i]<<" "<<result[i]<<endl;
    }
}
int main()
{
    int z,q,s,l;
    for(z=10;z<=50;z+=10)
    {
        for(q=10;q<=50;q+=10)
        {
            for(s=10;s<=50;s+=10)
            {
                for(l=10;l<=50;l+=10)
                {
                    if(z+q==s+l&&z+l>s+q&&z+s<q&&z!=q!=s!=l)
                    {
                        sort(z,q,s,l);
                    }
                }
            }
        }
    }
    return 0;
}
```

126.点评赛车

```
#include<stdio.h>
int main()
{
    int car;
    int a[4];
    int i;
    for(car=1;car<5;car++)
    {
        a[0]=(car==2);
        a[1]=(car==4);
        a[2]=(car!=3);
        a[3]=(car!=4);
        if((a[0]+a[1]+a[2]+a[3])==1)
        {
            printf("%d\n",car);
            for(i=0;i<4;i++)
            {
                if(a[i]) printf("%c\n",'A'+i);
            }
        }
    }
    return 0;
}
```

127.POJ2282 计数问题

```
#include <iostream>
#include <string.h>
#include<stdio.h>
using namespace std;
int a,b;
int ansa[10],ansb[10];
```

```
void count_digits(int s,int ans[],int times=1)//
{
	int i,d,p;
	if (s <= 0)return ;
	d = s % 10;
	p = s / 10;
	for (i = 1;i <= d;i ++)ans[i] += times;
	while(p > 0)
	{
		ans[p % 10] += (d + 1) * times;
		p = p / 10;
	}

	for (i = 0;i <= 9;i ++)ans[i] += times * (s / 10);

	times *= 10;
	count_digits((s / 10)-1,ans,times);
	return ;
}
int main()
{
	int i,j;
while(scanf("%d%d",&a,&b) != EOF && a + b > 0)
{
    memset(ansb,0,sizeof(ansb));
    memset(ansa,0,sizeof(ansa));
    if (a >= b)
    {
        swap(a,b);
    }
    a --;
    if (b > a)
    {
        count_digits(b,ansb);
        count_digits(a,ansa);
    }
    for(i=0;i<10;i++)
    {
        printf("%d%c",ansb[i]-ansa[i],i==9?'\n':' ');
    }

}
return 0;
}
```

128.POJ3974 求最长回文子串

```
#include<stdio.h>
#include<string.h>
#define M 1000008
char o[M],s[M<<1];
int p[M<<1];
int r;
int MMin(int a,int b)
{
	return a>b?b:a;
}
void Manacher()
{
	int i,j,max;
```

```
	int n=strlen(o);
	memset(s,'#',sizeof(s));
	for(i=0;i<n;i++)
		s[(i+1)<<1]=o[i];
	s[n=(n+1)<<1]=0;
	r=max=0;
	for(i=0;i<n;i++){
		if(max>i) p[i]=MMin(p[2*j-i],max-i);
		else p[i]=1;
		while(s[i+p[i]]==s[i-p[i]])
			p[i]++;
		if(p[i]>r) r=p[i];

		if(i+p[i]>max)
			max=i+p[i],j=i;
	}
}
int main()
{
	int t=0;
	~scanf("%s",o),o[0]^'E';
		Manacher();
		printf("%d",r-1);
	return 0;
}
```

129.涂色

```
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<cstring>
using namespace std;
int N,f[60][60];
char ch[100];
int main()
{
	scanf("%s",ch); N=strlen(ch);
	memset(f,123,sizeof f);
	for(int i=0;i<=N;++i)
	   f[i][i]=1,f[i+1][i]=0;
	for(int l=1;l<N;++l)
	  for(int i=0;i<N-l;++i)
	    {
		int j=i+l;
		f[i][j]=min(f[i+1][j]+1,f[i][j-1]+1);
		if(ch[i]==ch[j])
		   f[i][j]=min(f[i+1][j-1]+1,min(f[i+1][j],f[i][j-1]));
		for(int k=i;k<j;++k)
		   f[i][j]=min(f[i][j],f[i][k]+f[k+1][j]);
	    }
	printf("%d\n",f[0][N-1]);
	return 0;
}
```

130.子串替换

```
#include <stdio.h>
#include <string.h>
int main()
{
	char str[256],sub[256],rep[256],temp[256];
```

```
	char* p;
	gets(str);
	gets(sub);
	gets(rep);
	p = strstr(str,sub);	//从字符串 str1 中查找是否有字符串 str2，如果有，从 str1 中的 str2
位置起，返回 str1 中 str2 起始位置的指针，如果没有，返回 null
	if(p!=NULL){
	*p = 0;
	strcpy(temp, str);
	strcat(temp, rep);
	strcat(temp, p+strlen(sub));
	strcpy(str, temp);
	}
	printf("%s\n", str);
	return 0;
}
```

131.出现次数最多的字母

```
#include<stdio.h>
#include<string.h>
char s[280];
int num[30];
int main(void)
{
	int i,j,k;
	int n;
	char c;
	scanf("%d",&n);
	while(n--)
	{
		memset(num,0,sizeof(num));
		while(1)
		{
			c=getchar();
			if(c>='a'&&c<='z'||c>='A'&&c<='Z')
				break;
		}
		i=0;
		while(c!='\n')
		{
			s[i++]=c;
			c=getchar();
		}
		s[i]='\0';
		for(i=0;i<strlen(s);i++)
		{
			if(s[i]>='a'&&s[i]<='z')
				num[s[i]-'a'+1]++;
			if(s[i]>='A'&&s[i]<='Z')
				num[s[i]-'A'+1]++;
		}
		int max=0;
		for(i=1;i<=26;i++)
		{
			if(num[i]>max)
				max=num[i];
		}
		for(i=1;i<=26;i++)
		{
```

```
                if(num[i]==max)
                    printf("%c",'a'-1+i);
            }
            printf("\n");
        }
        return 0;
}
```

132.单词排序

```
#include<stdio.h>
#include<string.h>
char s[101][100];
char temp[100];
int main(void)
{
        int i,j,k;
        i=0;
        while(scanf("%s",s[i++])==1)
            ;
        for(j=0;j<i;j++)
        {
            for(k=j+1;k<i;k++)
            {
                if(strcmp(s[j],s[k])==1)
                {
                    strcpy(temp,s[j]);
                    strcpy(s[j],s[k]);
                    strcpy(s[k],temp);
                }
            }
        }
        printf("%s\n",s[1]);
        for(j=2;j<i;j++)
        {
            if(strcmp(s[j-1],s[j])==0)
                continue;
            printf("%s\n",s[j]);
        }
        return 0;
}
```

133.求序列中的众数

```
#include<stdio.h>
#include<math.h>
#include<string.h>
int main()
{
        int N,i,j,b[128],c,d,f[128];
        char a[128][100],e[100],h[128][100];
        scanf("%d",&N);
        for(i=0;i<N;i++)
        {
            scanf("%s",a[i]);
            while((a[i][0]=='0'||a[i][0]=='+')&&strlen(a[i])!=1)
            {
                for(j=0;j<=strlen(a[i])-1;j++)
                    a[i][j]=a[i][j+1];
            }
            while(a[i][0]=='-'&&a[i][1]=='0')
            {
```

```
                for(j=1;j<=strlen(a[i])-1;j++)
                    a[i][j]=a[i][j+1];
            }
            if(a[i][0]=='-'&&a[i][1]==0)        //-0 也是 0
                a[i][0]='0';
        }
        for(i=0;i<N;i++)
        {
            b[i]=0;
            f[i]=0;
        }
        for(i=0;i<N;i++)
        {
            for(j=i;j<N;j++)
            {
                if(strcmp(a[i],a[j])==0)
                {
                    b[i]++;
                    f[i]++;
                }
                strcpy(h[i],a[i]);
            }
        }
        for(i=0;i<N-1;i++)
        {
            for(j=i;j<N-1-i;j++)
            {
                if(b[j]>b[j+1])
                {
                    c=b[j];
                    b[j]=b[j+1];
                    b[j+1]=c;
                }
            }
        }
        if(b[N-1]==N)
            printf("no");
        else
        {
            for(i=0;i<N;i++)
            {
                if(f[i]==b[N-1])
                {
                    printf("%s",h[i]);
                    break;
                }
            }
        }
        scanf("%d",&N);
        return 0;
}
```

134.1 的个数

```
#include<stdio.h>
int main(void)
{
    int i,j,k,num;
    int n;
    scanf("%d",&n);
```

```
	while(n--)
	{
		num=0;
		scanf("%d",&k);
		while(k>1)
		{
			if(k%2==1)
				num++;
			k/=2;
		}
		if(k==1)
			num++;
		printf("%d\n",num);
	}
	return 0;
}
```

135.站队

```
#include<stdio.h>
int main()
{
 int i,j,k,l,m,n;
 for(i = 1; i <= 6; i++)
   for(j = 1; j <= 6; j++)
     for(k = 1; k <= 6; k++)
       for(l = 1; l <= 6; l++)
         for(m = 1; m <= 6; m++)
           for(n = 1; n <= 6; n++)
            if(i != 1 && j - i == 1 && l - k == 1
             && j - m == 2 && k - n == 4 && i != k
             &&i != l && i != m && i != n
             &&j != k &&j != l && j != m
             && j != n && k != l&& k != n&&
             k != m && l != m&& l != n&& m != n)
             printf("A at line %d\nB at line %d\nC at line %d\nD at line %d\nE at line %d\nF at
line %d",i,j,k,l,m,n);
	return 0;
}
```

136.数字谜

```
#include<stdio.h>
int main(void)
{
	int c,e,g,i,n,q,r,t=1,v;
	int j,k=0;
	for(c=2;c<=9;c++)
	for(v=1;v<=9;v++)
	if(c!=v&&v!=1)
	for(e=0;e<=9;e++)
	if(c!=e&&v!=e&&e!=1)
	for(g=0;g<=9;g++)
	if(c!=g&&v!=g&&e!=g&&g!=1)
	for(i=0;i<=9;i++)
	if(c!=i&&v!=i&&e!=i&&g!=i&&i!=1)
	for(n=0;n<=9;n++)
	if(c!=n&&v!=n&&e!=n&&g!=n&&i!=n&&n!=1)
	for(q=0;q<=9;q++)
	if(c!=q&&v!=q&&e!=q&&g!=q&&i!=q&&n!=q&&q!=1)
	for(r=0;r<=9;r++)
	if(c!=r&&v!=r&&e!=r&&g!=r&&i!=r&&n!=r&&q!=r&&r!=1)
```

```
    if((2*q+1)%10==e&&((2*q+1)/10+g+2*n)==21&&(i==4||i==9)&&((2*i+n+2)/10+i+2*c)%
10==e&&(((2*i+n+2)/10+i+2*c)/10+v)%10==r&&(((2*i+n+2)/10+i+2*c)/10+v)/10==1)
    printf("C:%d E:%d G:%d I:%d N:%d Q:%d R:%d T:%d V:%d",c,e,g,i,n,q,r,t,v);
    return 0;
}
```

138.全排列

```
#include<stdio.h>
#include<string.h>
char s[10];
int main(void)
{
    int i,j,k,l,m,n,wei;
    char a,b,c,d,e,f;
    char temp;
    scanf("%s",s);
    wei=strlen(s);
    for(i=0;i<wei-1;i++)
    {
        for(j=i+1;j<wei;j++)
        {
            if(s[i]>s[j])
            {
                temp=s[i];
                s[i]=s[j];
                s[j]=temp;
            }
        }
    }
    if(wei==1)
    {
        printf("%c",s[0]);
    }
    else if(wei==2)
    {
        printf("%c%c\n",s[0],s[1]);
        printf("%c%c",s[1],s[0]);
    }
    else if(wei==3)
    {
        for(i=0;i<=2;i++)
        {
            for(j=0;j<=2;j++)
            {
                for(k=0;k<=2;k++)
                {
                    a=s[i];b=s[j];c=s[k];
                    if(a!=b&&a!=c&&b!=c)
                        printf("%c%c%c\n",a,b,c);
                }
            }
        }
    }
    else if(wei==4)
    {
        for(i=0;i<=3;i++)
        {
            for(j=0;j<=3;j++)
            {
```

```
                    for(k=0;k<=3;k++)
                    {
                        for(l=0;l<=3;l++)
                        {
                            a=s[i];b=s[j];c=s[k];d=s[l];
                            if(a!=b&&a!=c&&a!=d&&b!=c&&b!=d&&c!=d)
                                printf("%c%c%c%c\n",a,b,c,d);
                        }
                    }
                }
            }
        }
        else if(wei==5)
        {
            for(i=0;i<=4;i++)
            {
                for(j=0;j<=4;j++)
                {
                    for(k=0;k<=4;k++)
                    {
                        for(l=0;l<=4;l++)
                        {
                            for(m=0;m<=4;m++)
                            {
                                a=s[i];b=s[j];c=s[k];d=s[l];e=s[m];

        if(a!=b&&a!=c&&a!=d&&a!=e&&b!=c&&b!=d&&b!=e&&c!=d&&c!=e&&d!=e)
                                printf("%c%c%c%c%c\n",a,b,c,d,e);
                            }
                        }
                    }
                }
            }
        }
        else if(wei==6)
        {
            for(i=0;i<=5;i++)
            {
                for(j=0;j<=5;j++)
                {
                    for(k=0;k<=5;k++)
                    {
                        for(l=0;l<=5;l++)
                        {
                            for(m=0;m<=5;m++)
                            {
                                for(n=0;n<=5;n++)
                                {
                                    a=s[i];b=s[j];c=s[k];d=s[l];e=s[m];f=s[n];

        if(a!=b&&a!=c&&a!=d&&a!=e&&a!=f&&b!=c&&b!=d&&b!=e&&b!=f&&c!=d&&c!=e&
&c!=f&&d!=e&&d!=f&&e!=f)
                                        printf("%c%c%c%c%c%c\n",a,b,c,d,e,f);
                                }
                            }
                        }
                    }
                }
```

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

139.1049 跳绳游戏

```
#include<stdio.h>
#include<string.h>
int a[100];
int main(void)
{
    int i,j,k,t,num;
    int n,m;
    scanf("%d",&n);
    while(n--)
    {
        memset(a,0,sizeof(a));
        scanf("%d",&m);
        for(i=1;i<=m;i++)
            scanf("%d",&a[i]);
        for(t=1,i=1,num=1;;)
        {
            if(num==a[i]&&i<=m&&a[i]<60)
            {
                t+=3;
                i++;
            }
            if(t>=60)
            {
                printf("%d\n",num);
                break;
            }
            else
            {
                t++;
                num++;
            }
        }
    }
    return 0;
}
```

140. 1081 字符串判等

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
char a[101];
char b[101];
char a1[101];
char b1[101];
char s[100];
void f()
{
    int i,j,k;
    for(i=0,j=0;i<strlen(a);i++)
    {
        if(isalpha(a[i]))
            a1[j++]=tolower(a[i]);
    }
    a1[j]='\0';
```

```
    for(i=0,k=0;i<strlen(b);i++)
    {
        if(isalpha(b[i]))
            b1[k++]=tolower(b[i]);
    }
    b1[k]='\0';
}
int main(void)
{
    //freopen("in.txt","r",stdin);
    int i,j,k;
    int n;
    char c;
    scanf("%d",&n);
    c=getchar();
    while(n--)
    {
        gets(a);
        gets(b);
        gets(s);
        f();
        if(strcmp(a1,b1)==0)
            printf("YES\n");
        else
            printf("NO\n");
    }
    return 0;
}
```

141.拨钟问题

```
#include <iostream>
#include <cstdio>
using namespace std;
int state[5][5] = {0};
int s[5][5] = {0};
int swit[5][5] = {0};
int cop[5][5] = {0};
int sum = 0;
int mi = 100;
void changeswit(int a, int b, int t)
{
    int no = (a - 1) * 3 + b;
    if(b == 2 && a != 2)
    {
        state[a][b - 1] = (state[a][b - 1] + t) % 4;
        state[a][b] = (state[a][b] + t) % 4;
        state[a][b + 1] = (state[a][b + 1] + t) % 4;
    }
    else if(a == 2 && b != 2)
    {
        state[a - 1][b] = (state[a - 1][b] + t) % 4;
        state[a][b] = (state[a][b] + t) % 4;
        state[a + 1][b] = (state[a + 1][b] + t) % 4;
    }
    else if(a == 2 && b == 2)
    {
        state[a][b] = (state[a][b] + t) % 4;
        state[a - 1][b] = (state[a - 1][b] + t) % 4;
        state[a + 1][b] = (state[a + 1][b] + t) % 4;
```

```
            state[a][b - 1] = (state[a][b - 1] + t) % 4;
            state[a][b + 1] = (state[a][b + 1] + t) % 4;
        }
        else
        {
            int dx[3] = {-1, 0, 1};
            int dy[3] = {-1, 0, 1};
            for(int i = 0; i < 3; i++)
                for(int j = 0; j < 3; j++)
                    state[a+dx[i]][b+dy[j]] = (state[a+dx[i]][b+dy[j]] + t) % 4;
        }
        return;

}
int main()
{
        for(int i = 1; i <= 3; i++)
            for(int j = 1; j <= 3; j++)
                scanf("%d",&s[i][j]);
        for(int i = 0; i < 4; i++)
        {
            for(int j = 0; j < 4; j++)
            {
                for(int k = 0; k < 4; k++)
                {
                    for(int l = 1; l < 4; l++)
                        for(int m = 1; m < 4; m++)
                        {
                            state[l][m] = s[l][m];
                            swit[l][m] = 0;
                        }
                    sum = 0;
                    swit[1][1] = i;
                    swit[1][2] = j;
                    swit[1][3] = k;
                    changeswit(1,1,i);
                    changeswit(1,2,j);
                    changeswit(1,3,k);
                    if(state[1][1] != 0)
                    {
                        swit[2][1] = 4 - state[1][1];
                        changeswit(2,1,4 - state[1][1]);
                    }
                    if(state[1][3] != 0)
                    {
                        swit[2][3] = 4 - state[1][3];
                        changeswit(2,3,4 - state[1][3]);
                    }
                    if(state[1][2] != 0)
                    {
                        swit[2][2] = 4 - state[1][2];
                        changeswit(2,2,4 - state[1][2]);
                    }
                    if(state[2][1] != 0)
                    {
                        swit[3][1] = 4 - state[2][1];
                        changeswit(3,1,4 - state[2][1]);
                    }
```

```
                    if(state[2][3] != 0)
                    {
                        swit[3][3] = 4 - state[2][3];
                        changeswit(3,3,4 - state[2][3]);
                    }
                    if(state[2][2] != 0) continue;
                    if(state[3][1] == state[3][2] && state[3][1] == state[3][3])
                    {
                        swit[3][2] = (4 - state[3][1]) % 4;
                        for(int l = 1; l < 4; l++)
                            for(int m = 1; m < 4; m++)
                                sum += swit[l][m];
                        if(sum < mi)
                        {
                            mi = sum;
                            for(int l = 1; l < 4; l++)
                                for(int m = 1; m < 4; m++)
                                    cop[l][m] = swit[l][m];
                        }
                    }
                    else continue;
                }
            }
        }
        bool f = 0;
        for(int i = 1; i < 4; i++)
            for(int j = 1; j < 4; j++)
            {
                while(cop[i][j]--)
                {
                    if(f) printf(" ");
                    printf("%d", 3*(i - 1) + j);
                    f = 1;
                }
            }
        printf("\n");
        return 0;
}
```

142.矩阵乘方和

```
#include <iostream>
#define MAX_SIZE 30
#include<stdio.h>
using namespace std;
int n, k, m;
struct MATRIX
{
    int val[MAX_SIZE][MAX_SIZE];

    MATRIX operator+(const MATRIX&l)
    {
        MATRIX temp;

        for(int i=0; i<n; i++)
            for(int j=0; j<n; j++)
                temp.val[i][j]=(val[i][j]+l.val[i][j])%m;
        return temp;
    }
    MATRIX operator*(const MATRIX&l)
```

```
	{
		MATRIX temp;

		for(int i=0; i<n; i++)
		{
			for(int j=0; j<n; j++)
			{
				temp.val[i][j] = 0;
				for(int k=0; k<n; k++)
					temp.val[i][j] = (temp.val[i][j] + val[i][k]*l.val[k][j]) % m;
			}
		}

		return temp;
	}
}A,S;
MATRIX power(MATRIX l, int k)
{
	MATRIX temp;

	if(k==1) return l;
	temp = l*l;
	if(k&1) return power(temp,(k-1)/2)*l;
	else	return power(temp,k/2);
}
MATRIX sum(int k)
{
	MATRIX	temp, tpow;
	if(k==1)
		return A;

	temp=sum(k/2);
	if(k&1)
	{
		tpow = power(A,k/2+1);
		temp = temp + temp*tpow + tpow;
	}
	else
	{
		temp = temp + power(A,k/2)*temp;
	}
	return temp;

}
int main()
{
	scanf("%d%d%d",&n,&k,&m);
	for(int i=0; i<n; i++)
	{
		for(int j=0; j<n; j++)
		{
			scanf("%d",&A.val[i][j]);
		}
	}
	S = sum(k);
	for(int i=0; i<n; i++)
	{
		for(int j=0; j<n-1; j++)
```

```
            {
                printf("%d ",S.val[i][j]);
            }
            printf("%d \n", S.val[i][n-1]);
        }
        return 0;
}
```

143.计算并输出杨辉三角形的前 n 行

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main()
{
        int n;
        int i,j,a[33][33];
        scanf("%d",&n);
            for(i=1;i<=n;i++)
            {
               a[i][1]=1;
               a[i][i]=1;
             }
            for (i=3;i<=n;i++)
              {
                 for(j=2;j<=i-1;j++)
                   {
                          a[i][j]=a[i-1][j-1]+a[i-1][j];
                    }
              }
            for(i=1;i<=n;i++)
                    {
                        for (j=1;j<i;j++)
                                printf("%d ",a[i][j]);
                        printf("%d\n",a[i][j]);
                    }
    return 0;
}
```

144.能种金币的聚宝盆

```
#include<stdio.h>
int main() {
        int n,i;
        scanf("%d",&n);
        int year(int,int,int);
        for(i=0; i<n; i++) {
            int y=0;
            int s,m;
            scanf("%d %d",&s,&m);
            printf("%d\n",year(s,m,y) - 1);
        }
}
int year(int s,int m,int y) {
        if(s>1000)
            s=1000;
        s=2*(s-m);
        m += (m*5+99)/100;
        y++;
        if(int(s)>0)
            y = year(s,m,y);
        return y;
```

```
}
```
145.统计单词
```
#include<stdio.h>
#include<string.h>
char s[1000][1000];
int dp[1000];
int main(void)
{
	int i=0,j=0,k;
	char c;
	for(k=0;k<1000;k++)
		dp[k]=1;
	while((c=getchar())!='\n')
	{
		if(c>='a'&&c<='z'||c>='A'&&c<='Z'||c=='\'')
		{
			s[i][j++]=c;
		}
		else if(j>0)
		{
			s[i][j]='\0';
			i++;
			j=0;
		}
	}
	for(k=0;k<i-1;k++)
	{
		for(j=k+1;j<i;j++)
		{
			if(strcmp(s[k],s[j])==0&&dp[j]!=0)
			{
				dp[k]++;
				dp[j]=0;
			}
		}
	}
	for(k=0;k<i;k++)
	{
		if(dp[k]>0)
		{
			printf("%s %d\n",s[k],dp[k]);
		}
	}
	return 0;
}
```
146.神奇的口袋(2)
```
#include<stdio.h>
#include<string.h>
int a[500];
long long dp[500][500];		//dp[i][j]表示从第 i 个数开始，要取出 j 体积的方法总数
int n;
int main(void)
{
	int i,j,k;
	long long count=0;
	scanf("%d",&n);
	for(i=1;i<=n;i++)
		scanf("%d",&a[i]);
```

```
    for(i=1;i<=n;i++)
        memset(dp[i],0,sizeof(dp[i]));
    for(i=1;i<=n;i++)
        dp[i][0]=0;
    dp[n][a[n]]=1;
    for(i=n-1;i>=1;i--)
    {
        for(j=1;j<=400;j++)
        {
            if(j-a[i]==0)
                dp[i][j]+=1;
            for(k=i+1;k<=n;k++)
            {
                if(j-a[i]>0)
                    dp[i][j]+=dp[k][j-a[i]];
            }
            dp[i][j]%=10000;
        }
    }
    for(i=1;i<=n;i++)
        count+=dp[i][400];
    printf("%lld",count%10000);
    return 0;
}
```

147.古代密码

```
#include<iostream>
#include<string.h>
#include<algorithm>
using namespace std;
int main(void)
{
    int i;
    int cipher[26],clear[26];
    memset(cipher,0,sizeof(cipher));          //对长度为 sizeof(cipher)=int*26=104 的连续内存空间 cipher 全部元素初始化为 0
    memset(clear,0,sizeof(clear));
    string input;
    cin>>input;
    for(i=0;i<input.length();i++)
    {
        cipher[input[i]-'A']++;
    }
    cin>>input;
    for(i=0;i<input.length();i++)
    {
        clear[input[i]-'A']++;
    }
    sort(cipher,cipher+26);          // 标准库自带排序函数 sort(ip_start,ip_end) 对某连续的地址段的对象内容进行升序快排（从小到大），<algorithm>
    sort(clear,clear+26);            //亦即 sort(ip_start+m,ip_start+n)，其中为从 ip_start+m 开始（包括 ip_start+m）对第 n 到第 m 个对象进行排序
    for(i=0;i<26;i++)
        if(cipher[i]!=clear[i])
        {
            cout<<"NO"<<endl;
            return 0;
        }
    cout<<"YES"<<endl;
```

```
    return 0;
}
```
148.平板上色
```
#include<cstring>
#include<iostream>
using namespace std;
struct node{
    int x1,y1,x2,y2,color;
}G[15];
int n,ans,deg[15];//n 表区域的个数，ans 存储最终结果，deg 存储拓扑图中各点的入度
bool vis[15],m[15][15];//vis 用于标记是否访问过，m 表示各点之间的联系
void buildG()//建立拓扑图，用于确定优先级
{
    for(int i=0;i<n;++i)
        for(int j=0;j<n;++j)
            if(G[i].y2==G[j].y1&&!(G[i].x2<G[j].x1||G[j].x2<G[i].x1)){
                m[i][j]=1;//建立关系网
                deg[j]++;//入度+1
            }
}
void dfs(int dep,int cnt,int curC)//深度优先搜索
{
    if(cnt>=ans) return;//剪枝，假如当前取画笔次数已大于之前结果，直接返回
    if(dep==n){//到达解答树目标状态，保存 ans 值，因前面有 cnt>=ans 时退出，所以当前
cnt 较小
        ans=cnt;
        return;
    }
    int i,j;
    for(i=0;i<n;++i)
        if(!deg[i]&&!vis[i]){//找入度为 0 且还未着色的点开始，
            vis[i]=1;
            for(j=0;j<n;++j)
                if(m[i][j])
                    deg[j]--;//子节点入度减一
            if(G[i].color==curC) dfs(dep+1,cnt,curC);//与当前颜色符合，只需往下遍历
            else dfs(dep+1,cnt+1,G[i].color);//否则，换画笔及颜色
            for(j=0;j<n;++j)
                if(m[i][j])
                    deg[j]++;//递归结束后，还原其初始状态
            vis[i]=0;//标志取消
        }
}
int main()
{
    int i,j,T;
    cin>>T;
    while(T--)
    {
        cin>>n;
        for(i=0;i<n;++i)
            cin>>G[i].y1>>G[i].x1>>G[i].y2>>G[i].x2>>G[i].color;
        memset(m,0,sizeof(m));
        memset(deg,0,sizeof(deg));
        buildG();
        ans=15;
        dfs(0,0,0);//都从 0 开始
        cout<<ans<<endl;
```

```
    }
    return 0;
}
```

149.符号三角形

```
#include"iostream"
using namespace std;
typedef unsigned char uchar;
#include<string.h>
#include<stdio.h>
char cc[2]={'+','-'};     //便于输出
int n,                          //第一行符号总数
    half,                       //全部符号总数一半
    counter;                    //1 计数，即“-”号计数
uchar **p;                      //符号存储空间
long sum;                       //符合条件的三角形计数
void Backtrace(int t)     //t，第一行第 t 个符号
{
    int i, j;
    if( t > n )
        sum++;
    else
    {
        for(i=0; i<2; ++i)
        {
            p[1][t] = i;            //第一行第 t 个符号
            counter += i;           //“-”号统计,因为"+"的值是 0

            for(j=2; j<=t; ++j)   //当第一行符号>=2 时，可以运算出下面行的某些符号，j 可代表行号
            {
                p[j][t-j+1] = p[j-1][t-j+1]^p[j-1][t-j+2];//通过异或运算下行符号，t-j+1 确定的很巧
                counter += p[j][t-j+1];
            }
            if( (counter <= half) && ( t*(t+1)/2 - counter <= half) )
            {//若符号统计未超过半数，并且另一种符号也未超过半数，同时隐含两者必须相等才能结束
                Backtrace(t+1);             //在第一行增加下一个符号
            }
            //回溯，判断另一种符号情况     像是出栈一样，恢复所有对 counter 的操作
            for(j=2; j<=t; ++j)
            {
                counter -= p[j][t-j+1];
            }
            counter -= i;
        }
    }
}
int main()
{
    cin >> n;
    counter = 0;
    sum = 0;
    half = n*(n+1)/2;
    int i=0;

    if( half%2 == 0 )
    {//总数须为偶数，若为奇数则无解
```

```
            half /= 2;
            p = new uchar *[n+1];

            for(i=0; i<=n; ++i)
            {
                p[i] = new uchar[n+1];
                memset(p[i], 0, sizeof(uchar)*(n+1));
            }

            Backtrace(1);
            for(i=0; i<=n; ++i)     //删除二维动态数组的方法
            {
                delete[] p[i];
            }
            delete[] p;
        }
        printf("%d",sum);
        return 0;
}
```

150.判断四边形

```
#include<stdio.h>
int x[10],y[10];
int f(int a,int b,int c)
{
        int A=y[b]-y[a];
        int B=x[a]-x[b];
        int C=x[b]*y[a]-x[a]*y[b];
        return A*x[c]+B*y[c]+C;
}
int main(void)
{
        //freopen("in.txt","r",stdin);
        int i,j,k;
        while(scanf("%d%d%d%d%d%d%d%d",&x[1],&y[1],&x[2],&y[2],&x[3],&y[3],&x[4],&y[
4])==8)
        {

        if(f(1,2,3)<0&&f(1,2,4)<0&&f(2,3,4)<0&&f(2,3,1)<0&&f(3,4,1)<0&&f(3,4,2)<0&&f(4,1,2
)<0&&f(4,1,3)<0)
                printf("yes\n");
            else
if(f(1,2,3)>0&&f(1,2,4)>0&&f(2,3,4)>0&&f(2,3,1)>0&&f(3,4,1)>0&&f(3,4,2)>0&&f(4,1,2)>0&
&f(4,1,3)>0)
                printf("yes\n");
            else
                printf("no\n");
        }
        return 0;
}
```

151.最大乘积

```
#include<stdio.h>
int main(void)
{
        int i,j,k;
        int n,sum=0;;
        int num[1000];
        scanf("%d",&n);
        for(i=2,j=1;;i++)
```

```
	{
		sum+=i;
		if(sum>n)
			break;
		num[j++]=i;
	}
	k=n-(sum-i);
	if(k==0)
	{
		for(i=1;i<j-1;i++)
			printf("%d ",num[i]);
		printf("%d",num[i]);
	}
	else if(k>=1&&k<num[j-1])
	{
		for(i=j-1;k>0;i--)
		{
			num[i]++;
			k--;
		}
		for(i=1;i<j-1;i++)
			printf("%d ",num[i]);
		printf("%d",num[i]);
	}
	else if(k==num[j-1])
	{
		for(i=1;i<j-1;i++)
			num[i]++;
		num[i]+=2;
		for(i=1;i<j-1;i++)
			printf("%d ",num[i]);
		printf("%d",num[i]);
	}
	return 0;
}
```

152.活动选择问题

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
	int s;
	int e;
}num[100001];
int comp(const void *a,const void *b)
{
	return ((node *)a)->e-((node *)b)->e;
}
int main(void)
{
	//freopen("in.txt","r",stdin);
	int i,j,k,count=1;
	int n;
	scanf("%d",&n);
	for(i=0;i<n;i++)
		scanf("%d%d",&num[i].s,&num[i].e);
	qsort(num,n,sizeof(num[0]),comp);
	for(i=1,j=0;i<n;i++)
	{
```

```
            if(num[i].s>=num[j].e)
            {
                count++;
                j=i;
            }
        }
        printf("%d",count);
        return 0;
}
```

153.旅行售货商问题（回溯）

```
#include <iostream>
#include<stdio.h>
using namespace std;
int n;
int a[17][17];
int x[17];
int bestc;
int cc;
void backtrace ( int t )
{
        if( t >    n)
        {
            if( a[x[n]][x[1]] != 0    && ((cc+a[x[n]][x[1]]) < bestc || bestc == 0))
                bestc = cc+a[x[n]][x[1]];
            return;
        }
        else
        {
            for (int i = t ; i<= n ; i++)
            {
                if(a[x[t-1]][x[i]] != 0 && ((cc+a[x[t-1]][x[i]]) < bestc || bestc == 0))//a[t][i]
                {
                    swap(x[t],x[i]);//swap  和  cc  的顺序问题啊你妹！
                    cc +=    a[x[t-1]][x[t]] ;
                    backtrace(t+1);
                    cc -=    a[x[t-1]][x[t]];
                    swap(x[i],x[t]);
                }
            }
        }
}
int main()
{
        cin >> n;
        for(int i = 1; i <= n ; i ++)
            for(int j = 1 ; j <= n ; j ++)
                cin >> a[i][j];
        cc = 0;
        bestc = 0;
        for(int i = 1 ; i <= n ; i ++)
            x[i] = i;
        backtrace(2);
        cout << bestc << endl;
        return 0;
}
```

154.象棋比赛（dp）

状态：a[n+1][j][k] = max{a[n][j][k], a[n][j-1][k]+Contest[n+1][0], a[n][j][k-1]+Contest[n+1][1]}。

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

```
int dp[1001][16][16]; //dp[n][i][j]表示有 n 个人，其中 i 个人执红，j 个人执黑的最大能力值
int num[1001][2];         //存储每个选手的能力值
int max(int a,int b,int c)
{
      if(a>=b)
      {
            if(a>=c)       return a;
            else       return c;
      }
      else
      {
            if(c>=b)       return c;
            else       return b;
      }
}
int main(void)
{
      int i=1,j,k,l,m,n;
      while(scanf("%d%d",&num[i][0],&num[i][1])==2)
            i++;
      i=i-1;
      for(j=0;j<=i;j++)
            dp[i][0][0]=0;
      dp[1][1][0]=num[1][0];
      dp[1][0][1]=num[1][1];
      for(j=2;j<=i;j++)
      {
            for(int k=0;(k<=15&&k<=j);k++)
                  for(int l=0;(l<=15&&(l+k)<=j);l++)
                  {
                        int largest=dp[j-1][k][l];
                        if((k>0)&&largest<dp[j-1][k-1][l]+num[j][0])
                              largest=dp[j-1][k-1][l]+num[j][0];
                        if((l>0)&& largest<dp[j-1][k][l-1]+num[j][1])
                               largest=dp[j-1][k][l-1]+num[j][1];
                        dp[j][k][l]=largest;
                  }
      }
      printf("%d",dp[i][15][15]);
      return 0;
}
```

155.基因串

```
#include <iostream>
using namespace std;
int main()
{
 char s[50][3];
 char f[101];
 bool c[101][101][26];
 int n, i, j, k;
 cin >> n;
 for (i = 0; i < n; i++)
 {
  char temp[4];
  cin >> temp;
  s[i][0] = temp[0];
  s[i][1] = temp[1];
  s[i][2] = temp[2];
```

```
}
cin >> f;
i = 0;
for (i = 0; i < 101; i++)
{
  for (j = 0; j < 101; j++)
  {
    for (k = 0; k < 26; k++)
    {
      c[i][j][k] = false;
    }
  }
}
i = 0;
while (f[i] != '\0')
{
  c[i][i][f[i] - 'A'] = true;
  i++;
}

int len = i;
for (j = 1; j < len; j++)
{
  for (i = 0; j + i < len; i++)
  {
    int t;
    for (t = 0; t < n; t++)
    {
      if (c[i][i + j][s[t][0] - 'A'])
        continue;
      for (k = i; k < i + j; k++)
      {
        if (c[i][k][s[t][1] - 'A'] && c[k + 1][i + j][s[t][2] - 'A'])
        {
          c[i][i + j][s[t][0] - 'A'] = true;
        }
      }
    }
  }
}
int g[101][101] = {0};
for (i = 0; i < len; i++)
{
  for (j = i; j < len; j++)
  {
    if (c[i][j]['S' - 'A'])
    {
      g[i][j] = 1;
    }
    else
      g[i][j] = 100;
  }
}

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

```
      int sum = g[i][i + j];
      for (k = i; k < i + j; k++)
      {
        if (g[i][k] + g[k + 1][i + j] < sum)
          sum = g[i][k] + g[k + 1][i + j];
      }
      g[i][i + j] = sum;
    }
  }
  if (g[0][len - 1] >= 100)
    cout << "-1" << endl;
  else
    cout << g[0][len - 1] << endl;
  return 0;
}
```

156.最小覆盖（贪心）

```
#include <iostream>
#include <cstring>
#include <stdlib.h>
using namespace std;
struct In
{
	int L ;
	int R ;
}data[100000];
int count_ = 0;
int M ;
int func()
{
	int num = 1;
	if(data[0].L > 0 || count_ == 0)
		return 0;
	else
	{
		int tmpR = data[0].R;
		bool flag = false;
		int maxR = 0;
		if(tmpR >= M )
			return num;
		for(int i = 1 ; i < count_ ;i ++)
		{
			if(data[i].L<= 0)
			{
				if (tmpR < data[i].R)
				{
					tmpR = data[i].R;
				}
				continue;
			}
			//cout<<"tmpR"<<tmpR <<endl;
			//cout<<"i"<<i<<endl;
			if(data[i].L <= tmpR)
			{
				//cout<<"im here asoll" <<endl;
				//cout<<"flag" <<flag <<endl;
				if (tmpR >= M)
					break;
				if(flag == false)
```

```
                {
                    flag = true;
                    num++;
                    //cout<<"num"<<num<<endl;
                }
                if(data[i].R > maxR)
                    maxR = data[i].R;
            }
            else
            {
                if(flag == true)
                {
                    i -- ;
                    flag = false;
                    tmpR = maxR;
                    maxR = 0;
                }
                else
                    return 0 ;
            }
        }
        return num;
    }
}
int cmp( const void *a ,const void *b)
{
    return (*(In *)a).L > (*(In *)b).L? 1 : -1;
}
int main()
{
    int l,r;
    cin >> M ;
    while(cin >> l >> r)
    {
        if( l == 0 && r == 0)
            break;
        if( l >= M || r <= 0 )
            continue;
        else
        {
            data[count_].L = l ;
            data[count_++].R = r ;
        }
    }
    qsort(data,count_,sizeof(data[0]),cmp);

    int num = func();
    if(num == 0)
        cout<< "No solution"<<endl;
    else
        cout<<num<<endl;
    return 0;
}
```

157.圆柱体装箱（贪心）

```
#include<stdio.h>
int a[100000]={0};
int main(void)
{
```

```
    int i,j,k,max=0,count=0,ok;
    int n;
    int l;
    scanf("%d",&n);
    scanf("%d",&l);
    for(i=0;i<n;i++)
    {
        scanf("%d",&k);
        a[k]++;
        if(max<k)
            max=k;
    }
    for(i=max;i>=1;i--)
    {
        while(a[i]>=1)
        {
            ok=0;
            for(j=l-i;j>=1;j--)
            {
                if(a[j]>1)
                {
                    count++;
                    a[j]--;
                    a[i]--;
                    ok=1;
                    break;
                }
                else if(a[j]==1)
                {
                    if(i!=j)
                    {
                        count++;
                        a[j]--;
                        a[i]--;
                        ok=1;
                        break;
                    }
                }
            }
            if(ok==0)
            {
                count++;
                a[i]--;
            }
        }
    }
    printf("%d",count);
    return 0;
}
```

158.通信系统

```
#include <stdlib.h>
#include <iostream>
using namespace std;
struct SYS { int b, p; } sys[100][100];
int cmp(const void * a, const void * b)
{
  SYS * c = (SYS *) a;
  SYS * d = (SYS *) b;
```

```
  return c -> p - d -> p;
}
int main()
{
  cout.setf(ios_base::showpoint);
  cout.setf(ios_base::fixed);
  cout.precision(3);
  int n;
  cin >> n;
  while(cin >> n)
  {
    int m[100], b[10000], bc = 0;
    for(int i = 0; i < n; i++)
    {
      cin >> m[i];
      for(int j = 0; j < m[i]; j++)
      {
        cin >> sys[i][j].b >> sys[i][j].p;
        b[bc++] = sys[i][j].b;
      }
      qsort(sys[i], m[i], sizeof(SYS), cmp);
    }
    double max = 0;
    for(int k = 0; k < bc; k++)
    {
      int sum = 0;
      bool could = true;
      for(int i = 0; i < n; i++)
      {
        int j;
        for(j = 0; j < m[i]; j++)
          if(sys[i][j].b >= b[k])
          {
            sum += sys[i][j].p;
            break;
          }
          if(j == m[i])
          {
            could = false;
            break;
          }
      }
      if(could)
      {
        double temp = double(b[k]) / sum;
        max = max > temp ? max : temp;
      }
        //max >?= double(b[k]) / sum;
    }
    cout << max << endl;
  }
  return 0;
}
```

159.积木正方形

```
#include <iostream>
#include <vector>
using namespace std;
int s;          //s，表示要组成的正方形的边长
```

```
int n;          //n(1<=n<=16)，表示积木的个数
int side[11];     //side[i]用来保存边长为 i 的小正方形的个数，这里约定正方形的边长最大为
10
int cols[41];     //cols[i]表示第 i 列已经被占用的行数+1,即最小可用行起始
vector<int> result;
int backTrace(int a)
{
 int selectCol, minRow, i, j, flag;
 if (a == n) return 1;
 minRow = 41;
 for (i = 1; i <= s; ++i)
  if (cols[i] < minRow)
  {
   selectCol = i;
   minRow = cols[i];
  }
 for (i = 10; i > 0; --i)
 {
  if (side[i] && selectCol-1+i <= s && minRow-1+i <= s)
  {
   flag = 1;
   for (j = selectCol; j < selectCol + i; ++j)
    if (cols[j] > minRow)
    {
     flag = 0;
     break;
    }
   if (flag)
   {
    for (j = selectCol; j < selectCol + i; ++j) cols[j] += i;
    side[i]--;
    if (backTrace(a + 1)) return 1;
    else
    {
     for (j = selectCol; j < selectCol + i; ++j) cols[j] -= i;
     side[i]++;
    }
   }
  }
 }
 return 0;
}
int main()
{
 int cases = 0;
 int i, temp;
 cin>>cases;
 int area;
 while (cases > 0)
 {
  cases--;
  cin>>s>>n;
  memset(side, 0, sizeof(side));
  area = 0;
  for (i = 0; i < n; ++i)
  {
   cin>>temp;
   side[temp]++;
```

```
      area += temp * temp;
    }
    if (area != s * s)
      result.push_back(0);
    else
    {
      for (i = 1; i <= 40; ++i) cols[i] = 1;
      int rr = backTrace(0);
      result.push_back(rr);
    }
  }
  for (i = 0; i < result.size(); ++i)
    if (result[i]) cout<<"yes"<<endl;
    else cout<<"no"<<endl;
  return 0;
}
```

160.字符游戏

```
#include<stdio.h>
char s[100][100];
char dp[1000];
int max=0;
int r,c;
void f(int i,int j,int n)
{
	int x,y,z,ok=1;
	for(x=1;x<n;x++)
	{
		if(s[i][j]==dp[x])
		{
			ok=0;
			dp[n]='\0';
			break;
		}
	}
	if(ok==1&&n>max)
		max=n;
	if(ok==1)
	{
		if(i-1>=0)
		{
			dp[n+1]=s[i-1][j];
			f(i-1,j,n+1);
		}
		if(i+1<r)
		{
			dp[n+1]=s[i+1][j];
			f(i+1,j,n+1);
		}
		if(j-1>=0)
		{
			dp[n+1]=s[i][j-1];
			f(i,j-1,n+1);
		}
		if(j+1<c)
		{
			dp[n+1]=s[i][j+1];
			f(i,j+1,n+1);
		}
```

```
        }
    }
}
int main(void)
{
    int i,j,k;
    scanf("%d%d",&r,&c);
    for(i=0;i<r;i++)
        scanf("%s",s[i]);
    dp[1]=s[0][0];
    f(0,0,1);
    printf("%d",max);
}
```

161.翻转游戏

```
#include <iostream>
#include <queue>
using namespace std;
int step[65535];    //记录步骤
bool flag[65535];    //防止重复搜索
unsigned short qState[65535]; //搜索的状态，正好可以用一个 16 位的无符号短整形表示
int rear = 0; //队列尾指针
int top = 0; //队列头指针
///初始化：读入棋盘初始状态并把它转化为整数存入队列头，黑的位为 1 白的为 0
void Init()
{
    unsigned short temp = 0;
    char c;
    for(int i=0; i < 4; i++)
        for(int j = 0; j < 4; j++)
        {
            cin>>c;
            if('b' == c)
                temp |= (1<<(i*4+j));
        }
    qState[rear++] = temp;
    flag[temp]    = true;
}
///翻转一个棋子并按规则对齐周围棋子附加影响
unsigned short move(unsigned short state, int i)
{
    unsigned short temp=0;
    temp |= (1<<i);
    if((i+1)%4 != 0) //右，且不在最右边
        temp |= (1<<(i+1));
    if(i%4 != 0) //左，且不在最左边
        temp |= (1<<(i-1));
    if(i+4 < 16) //下
        temp |= (1<<(i+4));
    if(i-4 >= 0) //上
        temp |= (1<<(i-4));
    return (state ^ temp);
}
//广度优先搜索，从队列中循环取出状态，并把翻转 16 次（即所有情况），一旦发现满足要求的立即停止，否则加入队列
bool BFS()
{
    while(rear > top)
    {
        unsigned short state = qState[top++];
```

```
            //qState.pop();
            for(int i=0; i < 16; i++)
            {
                unsigned short temp;
                temp = move(state,i);
                if(0 == state || 65535 == state)
                {
                    cout<<step[state];
                    return true;
                }
                else if(!flag[temp]) //防止重复搜索
                {
                    //qState.push(temp);
                    qState[rear++] = temp;
                    flag[temp] = true;
                    step[temp] = step[state]+1;
                }
            }
        }
        return false;
}
int main(void)
{
        Init();
        if(!BFS()) cout<<"Impossible";
        char c;
        cin>>c;
}
```

162.cleaning robot

```
#include <stdio.h>
#include<string.h>
#define min(a,b) ((a)<(b)?(a):(b))
#define EMPTY 255
#define WALL 254
#define QR 1024000
#define MAX 0x7FFFFFFF
typedef struct str_a { int serial, x, y; } command;
int map[30][30];
short dyna[1024][30][30];
int dx[4] = {-1, 0, 1, 0};
int dy[4] = {0, -1, 0, 1};
command queue[QR]; int now, add;
int main ()
{
        int i, j, w, h, ans, dirt; char ar[30], a;
        int rx, ry, rs, step, rd, tx, ty, ts, cur, ro;

        while (1)
        {
                scanf("%d %d", &w, &h);
                if (w == 0 && h == 0) break;
                memset(dyna, MAX, sizeof(dyna));
                now = add = dirt = 0; ans = MAX;
                for (i = 0; i < h; i++)
                {
                        scanf("%s", ar);
                        for (j = 0; j < w; j++)
                        {
```

```
                    a = ar[j];
                    if (a == '.') map[i][j] = EMPTY;
                    else if (a == 'x') map[i][j] = WALL;
                    else if (a == 'o') map[i][j] = EMPTY, rx = i, ry = j;
                    else map[i][j] = dirt++;
                }
            }
            dyna[0][rx][ry] = 0;
            ro = (1 << dirt) - 1;
            queue[add].serial = 0, queue[add].x = rx, queue[add].y = ry; add++;
            while (now != add)
            {
                cur = now % QR;
                rx = queue[cur].x, ry = queue[cur].y, rs = queue[cur].serial; now++;
                step = dyna[rs][rx][ry];
                for (i = 0; i < 4; i++)
                {
                    tx = rx + dx[i];
                    ty = ry + dy[i]; ts = rs;
                    if (tx >= 0 && tx < h && ty >= 0 && ty < w)
                    {
                        rd = map[tx][ty];
                        if (rd == WALL) continue;
                        else if (rd != EMPTY) ts |= 1 << rd;
                        if (dyna[ts][tx][ty] < 0)
                        {
                            dyna[ts][tx][ty] = step + 1;
                            if (ts == ro) ans = min(ans, step + 1);
                            cur = add % QR;
                            queue[cur].serial = ts,
                            queue[cur].x = tx,
                            queue[cur].y = ty; add++;
                        }
                    }
                    else continue;
                }
            }
            if (ans == MAX) printf("%d\n", -1);
            else printf("%d\n", ans);
        }
        return 0;
}
```

163.布尔表达式

```
#include <stdio.h>
#include <string.h>
char a[1111] = {0}, s[1111], *p;
int b[1111], c[1111], f[222];
int i, n = 0, m;
int calc(int l, int r){
        while (c[l] == r) ++l, --r;
        if (l == r) return a[l];
        int i, j, p = l;
        for (i = l; i <= r; ++i) {
                if (f[a[p]] < f[a[i]] || (f[a[p]] == f[a[i]] && a[p] != '!')) p = i;
                if (a[i] == '(') i = c[i] - 1;
        }
        if (a[p] == '!') return !calc(p + 1, r);
        i = calc(l, p - 1), j = calc(p + 1, r);
```

```
    return a[p] == '&' ? i && j : i || j;
}
int main(){
    memset(f, 0, sizeof f);
    f['!'] = 1, f['&'] = 2, f['|'] = 3;
    while (p = gets(s)) {
        for (n = 0; *p; ++p)
            if (*p == 'F') a[++n] = 0;
            else if (*p == 'V') a[++n] = 1;
            else if (*p == '!') a[n] == '!' ? --n : a[++n] = '!';
            else if (*p != ' ') a[++n] = *p;
        memset(c, 0, sizeof c);
        for(m = 1, i = 1; i <= n; ++i)
            if (a[i] == '(') b[++m] = i;
            else if (a[i] == ')') c[i] = b[m], c[b[m--]] = i;
        puts(calc(1, n) ? "V" : "F");
    }
    return 0;
}
```

164.POJ1700 过河问题

很自然想到贪心，

一种思路是由于必须有人把船开回来，尽量把这个时间缩短，因此总是让最快的陪同最慢的过去，然后最快的驾船回来。但是这样每个比最快的慢的时间都要算进去。

于是想到可以让最慢的和次慢的一起走，这样虽然消耗了最慢的时间，但是不用再消耗次慢的时间。这样必须有人把船开回来，而且绝不能是次慢的这个，因此可以先让最快次快的去一次，让最快的回来，次快的留下预备用。这样相当于每次运最慢的之前首先进行一次预处理。比较这种开支和第一种思路开支的大小，选择合适的方案来进行一次慢运输。

```
#include<iostream>
#include<algorithm>
using namespace std;
int T,N,elapse,l[1005];
int main(){
    cin>>T;
    while(T--){
        cin>>N;
        for(int i=0;i<N;i++){
            cin>>l[i];
        }
        elapse=0;
        sort(l,l+N);
        int i=N-1;
        if(i==0)
            elapse+=l[0];
        while(i>0){
            if(i==1){
                elapse+=l[1];
            }else if(i==2){
                elapse+=(l[0]+l[1]+l[2]);
            }else{
                if(2*l[1]-l[i-1]-l[0]<0){
                    elapse+=(l[0]+2*l[1]+l[i]);
                }else{
                    elapse+=(2*l[0]+l[i-1]+l[i]);
                }
            }
            i=i-2;
        }
        cout<<elapse<<endl;
```

```
    }
    return 0;
}
```

165.取石子游戏

```
#include<stdio.h>
int main(void)
{
    int i,j,k,ok;
    int a,b;
    while(scanf("%d%d",&a,&b)==2)
    {
        ok=1;
        if(a==0&&b==0)
            break;
        if(a/b>=2||b/a>=2)
            printf("win\n");
        else
        {
            if(a==b)
                printf("win\n");
            else
            {
                j=0;
                while(a!=0&&b!=0)
                {
                    if(a/b>=2||b/a>=2)
                    {
                        ok=0;
                        if(j%2==1)
                        {
                            printf("lose\n");
                            break;
                        }
                        else
                        {
                            printf("win\n");
                            break;
                        }
                    }
                    else if(a>b)
                    {
                        a=a-b;
                        j++;
                    }
                    else
                    {
                        b=b-a;
                        j++;
                    }
                }
                if(j%2==0&&ok==1)
                    printf("win\n");
                else if(ok==1)
                    printf("lose\n");
            }
        }
    }
    return 0;
```

```
}
```
166.POJ1067 取石子游戏
```
#include <stdio.h>
int main ()
{
int a,b,c;
const double d=1.6180339887498948482045 ;
while(scanf("%d%d",&a,&b)!=EOF){
if (a==b) {printf("1\n"); continue;}
if (a>b) {c=a-b;a=b;}
else c=b-a;
if (a==(int)(c*d))
printf("0\n");
else printf("1\n");
}
return 0;
}
```
167.POJ1821 粉刷篱笆（dp）
```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
int N, K;
int f[101][16001];
struct node
{
	int len, price, sit;
	bool operator<(const node & a)const
	{
		return sit < a.sit;
	}
} worker[101];
void dp()
{
	int ans = 0;
	int Q[16001];
	for (int i = 1; i <= K; ++i)
	{
		int head = 1, tail = 0;
		for (int j = 0; j < worker[i].sit; ++j)
			f[i][j] = f[i - 1][j];
		for (int j = max(worker[i].sit - worker[i].len, 0); j < worker[i].sit; ++j)
		{
			while (head <= tail &&
				f[i - 1][Q[tail]] - Q[tail] * worker[i].price <= f[i - 1][j] - j *
worker[i].price)
				--tail;
			Q[++tail] = j;
		}
		for (int j = worker[i].sit; j <= N; ++j)
		{
			while (head <= tail && Q[head] < j - worker[i].len) ++head;
			f[i][j] = max(f[i - 1][j], f[i][j - 1]);
			if (head <= tail)
				f[i][j] = max(f[i][j], f[i - 1][Q[head]] + (j - Q[head]) * worker[i].price);
//			while (head <= tail &&
//				f[i - 1][Q[tail]] - Q[tail] * worker[i].price <= f[i - 1][j] - j *
worker[i].price)
```

```
//                          --tail;
//                  Q[++tail] = j;
            }
        }
        printf("%d\n", f[K][N]);
}
int main()
{
        scanf("%d%d", &N, &K);
        for (int i = 1; i <= K; ++i)
                scanf("%d%d%d", &worker[i].len, &worker[i].price, &worker[i].sit);
        sort(worker + 1, worker + K + 1);
        dp();
        return 0;
}
```

168.修复牛棚

```
#include<stdio.h>
#include<stdlib.h>
int num[1000];
int dp[1000];
int comp(const void *a,const void *b)
{
        return *(int *)b-*(int *)a;
}
int cop(const void *a,const void *b)
{
        return *(int *)a-*(int *)b;
}
int main(void)
{
        int i,j,k,sum;
        int m,s,c;
        scanf("%d%d%d",&m,&s,&c);
        for(i=0;i<c;i++)
                scanf("%d",&num[i]);
        qsort(num,c,sizeof(num[0]),cop);
        for(i=0;i<c-1;i++)
                dp[i]=num[i+1]-num[i]-1;
        qsort(dp,c-1,sizeof(dp[0]),comp);
        m--;
        sum=num[c-1]-num[0]+1;
        k=0;
        while(m--)
        {
                sum-=dp[k];
                k++;
        }
        printf("%d",sum);
        return 0;
}
```

169.连续邮资问题(回溯)

```
#include<cstring>
#include<cstdio>
#include<cstdlib>
#include<iostream>
#define INF 2147483647
#define MAXN 2000
using namespace std;
```

```
int h, k;
int ans[MAXN], maxStampVal, stampVal[MAXN], maxVal[MAXN], y[MAXN];
bool occur[MAXN];
void search(int cur){
	if(cur >= k){
		if(maxVal[cur-1] > maxStampVal){
			maxStampVal = maxVal[cur-1];
			memcpy(ans, stampVal, sizeof(stampVal));
		}
		return ;
	}
	int tmp[MAXN];
	memcpy(tmp, y, sizeof(y));
	for(int i=stampVal[cur-1]+1; i<=maxVal[cur-1]+1; ++i){

		stampVal[cur] = i;
		// 关键步骤，利用了动态规划的思想
		for(int j=0; j<stampVal[cur-1]*h; ++j)if(y[j]<h){
			for(int num=1; num<=h-y[j]; ++num){
				if(y[j]+num < y[j + i*num] && (j+i*num<MAXN))
					y[j+i*num] = y[j]+num;
			}
		}
		int r = maxVal[cur-1];
		while(y[r+1]<INF) r++;
		maxVal[cur] = r;
		search(cur+1);
		memcpy(y, tmp, sizeof(tmp));
	}
}
int main(){
	// h 数量， k 面额数
	while(scanf("%d %d", &k, &h), h+k){
		stampVal[0] = 1;
		maxVal[0] = h;
		int i;
		for(i=0; i<=h; ++i)
			y[i] = i;
		while(i<MAXN) y[i++] = INF;
		maxStampVal = -2147483645;
		search(1);
		printf("%d\n", maxStampVal);
	}
	return 0;
}
```

170.跳格问题

```
#include<stdio.h>
int gezi[30];
int main(void)
{
	int i,j,k,count=0;
	int n;
	scanf("%d",&n);
	gezi[0]=1;		//起点
	for(i=1;i<=n;i++)
		scanf("%d",&gezi[i]);
	i=0;
	while(i<n+1)
```

```
    {
        if(i<0)
            i=0;
        j=i;
        if(gezi[j]>0)
        {
            j+=gezi[i];
            gezi[i]=0;
            count++;
        }
        else if(gezi[j]==0)
        {
            count+=2;
            j++;
        }
        else if(gezi[j]<0)
        {
            j+=gezi[j];
            gezi[i]=0;
            count++;
        }
        i=j;
    }
    printf("%d",count);
    return 0;
}
```

171.木棍正方形

```
# include <iostream>
# include <vector>
# include <algorithm>
# include <queue>
using namespace std;
int EdgeLength = 0;
vector<int> stickStorage;
vector<bool> usage;
bool pathFinder(int stickRemain, int spaceRemain, int subscription)
{
    int spcR = spaceRemain;
    int stkR = stickRemain;
    if ( spaceRemain == 0 )
    {
        if ( stickRemain == 0 )
        {
            return true;
        }
        spcR = EdgeLength;
        subscription = 0;
    }
    int size = stickStorage.size();
    for ( int i = subscription ; i < size; i++ )
    {
        if ( usage[i] == false && stickStorage[i] <= spcR )
        {
            usage[i] = true;
            if ( pathFinder(stkR-1,spcR-stickStorage[i],i+1) )
            {
                return true;
            }
```

```
                usage[i] = false;
                if ( stickStorage[i] == spcR || spcR == EdgeLength )    //If the current stick fills a
line, there's no need to compare any more
                {
                        break;
                }
                while ( stickStorage[i] == stickStorage[i+1] && i < size-1 )
                {
                        i++;
                }
        }
    }
    return false;
}
int main()
{
    int n = 0;
    int stickNum = 0;
    int currentStickLen = 0;
    bool possible = false;
    int sum = 0;
    cin >> n;
    int j = 0;
    for ( int i = 0; i < n; i++ )
    {
        possible = false;
        EdgeLength = 0;
        sum = 0;
        stickStorage.clear();
        usage.clear();
        cin >> stickNum;
        for ( j = 0; j < stickNum; j++ )
        {
            cin >> currentStickLen;
            sum += currentStickLen;
            stickStorage.push_back(currentStickLen);
            usage.push_back(false);
        }
        sort(stickStorage.begin(),stickStorage.end(),greater<int>());
        if ( sum%4 != 0 )
        {
            possible = false;
        }
        else
        {
            EdgeLength = sum/4;
            possible = pathFinder(stickStorage.size(),EdgeLength,0);
        }
        if ( possible )
        {
            cout << "yes" << endl;
        }
        else
        {
            cout << "no" << endl;
        }
    }
    return 0;
```

```
}
```

172.POJ1681 画家问题

```
#include <iostream>
#include <cstdio>
#include <cmath>
using namespace std;
int t = 0;
int n = 0;
char col[20];
bool c[20][20] = {0};
bool colored[20][20] = {0};
bool swit[20][20] = {0};
void changeswit(int a, int b)
{
	colored[a][b] = !colored[a][b];
	colored[a - 1][b] = !colored[a - 1][b];
	colored[a + 1][b] = !colored[a + 1][b];
	colored[a][b - 1] = !colored[a][b - 1];
	colored[a][b + 1] = !colored[a][b + 1];
}
int main()
{
	scanf("%d", &t);
	for(int ti = 1; ti <= t; ++ti)
	{
		int sum = 0;
		scanf("%d", &n);
		int min = n * n + 1;
		for(int i = 1; i <= n; ++i)
		{
			scanf("%s", col);
			for(int j = 1; j <= n; ++j)
			{
				if(col[j - 1] == 'y') c[i][j] = 1;
				else c[i][j] = 0;
			}
		}
		for(int k = 0; k < pow(2.0, n); k++)
		{
			for(int i = 1; i <= n; i++)
				for(int j = 1; j <= n; j++)
				{
					colored[i][j] = c[i][j];
				}
			for(int i = 1; i <= n; i++)
			{
				swit[1][i] = (k>>(i - 1)) & 1;
				if(swit[1][i])
				{
					changeswit(1, i);
					sum++;
				}
			}
			for(int i = 2; i <= n; i++)
				for(int j = 1; j <= n; j++)
					if(!colored[i - 1][j])
					{
						swit[i][j] = 1;
```

```
                                sum++;
                                changeswit(i, j);
                            }
                    int mark = 0;
                    for(int j = 1; j <= n; j++)
                        if(!colored[n][j]) mark = 1;
                    if(!mark && sum < min) min = sum;
                    sum = 0;
                }
                if(min != n*n + 1) printf("%d\n",min);
                else printf("inf\n");
        }
        return 0;
}
```

173.木棒分割（贪心）

```
#include <iostream>
#include<queue>
using namespace std;
int main()
{
        priority_queue <int ,vector<int>,greater<int> >q;
        int num;
        int n;
        cin>>n;
        for(int i =0;i < n ;i ++)
        {
                cin >> num;
                q.push(num);
        }
        int a,b,tmp;
        int sum = 0;
        while(q.size()!=1)
        {
                a = q.top();
                q.pop();
                b = q.top();
                q.pop();
                tmp = a+b;
                q.push(tmp);
                //cout<<a << " " <<b <<endl;
                //cout<< "tmp" <<tmp <<endl;
                sum+= tmp;
        }
        cout << sum << endl;
        return 0;
}
```

174.佳佳的筷子（dp）

```
#include <iostream>
using namespace std;
#include<stdio.h>
#include<stdlib.h>
#define min(a,b) (a)<(b)?(a):(b)
#define sqr(a) (a)*(a)
#define MAXK 1100
#define MAXN 5100
int comp(const void *a,const void *b)
{
        return *(int *)a-*(int *)b;
```

```
}
int t,T,i,j,k,n,a[MAXN],f[MAXN][MAXK],temp;
int main()
{
	cin>>n>>k;
	for (i=1;i<=n;i++) cin>>a[i];
	a[0]=0;
	qsort(a,n+1,sizeof(a[0]),comp);
	for (i=1;i<=n/2;i++)
	{
		temp=a[n+1-i];a[n+1-i]=a[i];a[i]=temp;
	}
	for (i=0;i<=n;i++) f[i][0]=0;
	for (j=1;j<=k;j++)
	{
		f[j*3][j]=f[j*3-2][j-1]+sqr(a[j*3]-a[j*3-1]);
		for (i=j*3+1;i<=n;i++)
			f[i][j]=min(f[i-1][j],f[i-2][j-1]+sqr(a[i]-a[i-1]));
	}
	cout<<f[n][k]<<endl;
    return 0;
}
```

175.球桌出租（dp）

```
#include<iostream>
#include<stdio.h>
#include<stdlib.h>
#include<list>
#include<algorithm>
using namespace std;
int N = 0;
struct bill
{
    int s;
    int t;
    int c;
}tax[100002];
long long int save[100002];
int cmp(const void *A,const void *B)
{
    if((*(bill*)A).t == (*(bill*)B).t)
    {
        return (*(bill*)A).s > (*(bill*)B).s;
    }
    else
    return (*(bill*)A).t > (*(bill*)B).t;
}
int main()
{
    cin>>N;
    list<bill>L;
    for(int i = 0;i<N;i++)
        scanf("%d%d%d",&tax[i].s,&tax[i].t,&tax[i].c);
    qsort(tax,N,sizeof(tax[0]),cmp);
    save[0] = tax[0].c;
    long long int max = save[0];
    for(int i = 1;i<N;i++)
    {
        int j = i - 1;
```

```
            int l = 0 ,r = j,mid;
            while(r - l > 1)
            {
                mid = (r+l)/2;
                if(tax[mid].t <= tax[i].s)
                    l = mid;
                else
                    r = mid -1;
            }
            if(tax[l].t>tax[i].s && tax[l+1].t>tax[i].s)
            save[i] = tax[i].c;
            else
            {
                if(tax[l+1].t<=tax[i].s)
                    save[i] = tax[i].c + save[l+1];
                else
                    save[i] = tax[i].c + save[l];
            }
            if(save[i]< save[i - 1])
                save[i] = save[i - 1];
            if(save[i] > max)
                max = save[i];
        }
        cout<<max;
        return 0;
}
```

176.天平（dp）

首先定义一个平衡度 j 的概念

当平衡度 j=0 时，说明天枰达到平衡，j>0，说明天枰倾向右边（x 轴右半轴），j<0 则相反

那么此时可以把平衡度 j 看做为衡量当前天枰状态的一个值

因此可以定义一个 状态数组 dp[i][j]，意为在挂满前 i 个钩码时，平衡度为 j 的挂法的数量。

由于距离 c[i]的范围是-15~15，钩码重量的范围是 1~25，钩码数量最大是 20

因此最极端的平衡度是所有物体都挂在最远端，因此平衡度最大值为 j=15*20*25=7500。原则上就应该有 dp[ 1~20 ][-7500 ~ 7500 ]。

因此为了不让下标出现负数，做一个处理，使使得数组开为 dp[1~20][0~15000]，则当 j=7500 时天枰为平衡状态。那么每次挂上一个钩码后，对平衡状态的影响因素就是每个钩码的力臂力臂=重量 *臂长 = w[i]*c[k]那么若在挂上第 i 个砝码之前，天枰的平衡度为 j(换言之把前 i-1 个钩码全部挂上天枰后，天枰的平衡度为 j)则挂上第 i 个钩码后，即把前 i 个钩码全部挂上天枰后，天枰的平衡度 j=j+ w[i]*c[k]。其中 c[k]为天枰上钩子的位置，代表第 i 个钩码挂在不同位置会产生不同的平衡度

不难想到，假设 dp[i-1][j] 的值已知，设 dp[i-1][j]=num（即已知把前 i-1 个钩码全部挂上天枰后得到状态 j 的方法有 num 次）那么 dp[i][ j+ w[i]*c[k] ] = dp[i-1][j] = num

(即以此为前提，在第 k 个钩子挂上第 i 个钩码后，得到状态 j+ w[i]*c[k]的方法也为 num 次)

想到这里，利用递归思想，不难得出 状态方程 dp[i][ j+ w[i]*c[k] ]= Σ（dp[i-1][j]）

有些前辈推导方式稍微有点不同，得到的 状态方程为 dp[i][j] =Σ(dp[i - 1][j - c[i] * w[i]])

其实两条方程是等价的，这个可以简单验证出来，而且若首先推导到第二条方程，也必须转化为第一条方程，这是为了避免下标出现负数

结论：

最终转化为 01 背包问题。状态方程 dp[i][ j+ w[i]*c[k] ]= Σ（dp[i-1][j]）

初始化：dp[0][7500] = 1;  //不挂任何重物时天枰平衡，此为一个方法

复杂度 O(C*G*15000)  完全可以接受

```
#include<stdio.h>
#include<string.h>
int dp[21][15001];//dp[i][j]表示前 i 个砝码达到 j 平衡度的挂法数量，0~15000，j=7500 为平衡位置
int weigh[21];      //砝码质量
int len[21];        //挂钩位置
int main(void)
```

```
{
	int i,j,k;
	int c,g;
	scanf("%d%d",&c,&g);
	for(i=1;i<=c;i++)
		scanf("%d",&len[i]);
	for(i=1;i<=g;i++)
		scanf("%d",&weigh[i]);
	for(i=0;i<=20;i++)
		memset(dp[i],0,sizeof(i));
	dp[0][7500]=1;
	for(i=1;i<=g;i++)
		for(j=0;j<=15000;j++)
			if(dp[i-1][j])
			for(k=1;k<=c;k++)
				dp[i][j+weigh[i]*len[k]]+=dp[i-1][j];
	printf("%d",dp[g][7500]);
	return 0;
}
```

177.路径计数

```
#include<iostream>
#include<math.h>
using namespace std;
unsigned comp(unsigned n,unsigned m)
{
	unsigned a=m+n;
	unsigned b=(m<n?m:n);
	double cnm=1.0;
	while(b>0)
		cnm*=(double)(a--)/(double)(b--);
	cnm+=0.5;		//double 转 unsigned 会强制截断小数，必须先四舍五入
	return (unsigned)cnm;
}
int main(void)
{
	unsigned m,n;
	while(true)
	{
		cin>>m>>n;
		if(!m && !n)//承认这题的猥琐吧！竟然有其中一边为 0 的矩阵，一定要&&，用||
会 WA
			break;
		cout<<comp(n,m)<<endl;
	}
	return 0;
}
```

178.爆竹与箱子（dp）

```
#include<stdio.h>
#include<string.h>
int dp[11][101][101]={0};	//dp[k][i][j]表示 k 个箱子在爆破区间[i,j]上在最坏情况下最少爆
竹数
int max(int a,int b)
{
	return (a>b)?a:b;
}
int main(void)
{
	int i,j,l,n,x,s;
```

```
    int k,m,t;
    for(i=1;i<=100;i++)
        for(j=i;j<=100;j++)
            dp[1][i][j]=(j-i+1)*(i+j)/2;
    for(i=2;i<=10;i++)
        for(j=1;j<=100;j++)
            dp[i][j][j]=j;
    for(i=2;i<=10;i++)          //盒子数
    {
        for(x=1;x<100;x++)     //上下限差
        {
            for(j=1;j<=100-x;j++)
            {
                dp[i][j][j+x]=10000000;
                for(s=j;s<=j+x;s++)
                {
                    if(max(s+dp[i-1][j][s-1],s+dp[i][s+1][j+x])<dp[i][j][j+x])
                        dp[i][j][j+x]=max(s+dp[i-1][j][s-1],s+dp[i][s+1][j+x]);
                }
            }
        }
    }
    scanf("%d",&t);
    while(t--)
    {
        scanf("%d%d",&k,&m);
        printf("%d\n",dp[k][1][m]);
    }
}
```

179.合唱队形（dp）

分析：向两端的最长升序子列。故设为 ldp 和 rdp。

```
#include<stdio.h>
int num[101];
int ldp[101]={0};
int rdp[101]={0};
int main(void)
{
    int i,j,k;
    int n;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
        scanf("%d",&num[i]);
    ldp[1]=0;
    rdp[n]=0;
    for(i=2;i<=n;i++)
    {
        k=0;
        for(j=1;j<i;j++)
        {
            if(num[i]>num[j])
            {
                if(k<ldp[j]+1)
                    k=ldp[j]+1;
            }
        }
        ldp[i]=k;
    }
    for(i=n-1;i>=1;i--)
```

```
    {
        k=0;
        for(j=n;j>i;j--)
        {
            if(num[i]>num[j])
            {
                if(k<rdp[j]+1)
                    k=rdp[j]+1;
            }
        }
        rdp[i]=k;
    }
    k=0;
    for(i=1;i<=n;i++)
        if(ldp[i]+rdp[i]>k)
            k=ldp[i]+rdp[i];
    printf("%d",n-k-1);
    return 0;
}
```

180.能量项链（dp）

```
#include <cstdio>
#include <algorithm>
using namespace std;
int n;
int l[210][210],r[210][210],s[210][210];
int ff()
{
    for (int p = 1; p < n; p ++)
        for (int i = 0; i < 2 * n - 1; i ++)
            if (i + p < 2 * n - 1)
            {
                int j = i + p;
                for (int k = i; k < j; k ++)
                {
                    s[i][j] = max(s[i][j], s[i][k] + s[k+1][j] + l[i][k] * r[i][k] * r[k+1][j]);
                    l[i][j] = l[i][k];
                    r[i][j] = r[k+1][j];
                }
            }
    int res = 0;
    for (int i = 0; i < n; i ++)
        if (s[i][i + n - 1] > res)
            res = s[i][i + n - 1];
    return res;
}
int main()
{
    scanf("%d",&n);
    for (int i = 0; i < n; i ++)
    {
        int q;
        scanf("%d",&q);
        l[i][i] = q;
        if (i != 0)
            r[i-1][i-1] = q;
        if (i == n - 1)
            r[i][i] = l[0][0];
    }
```

```
    for (int i = 0; i < n; i ++)
    {
        l[i+n][i+n] = l[i][i];
        r[i+n][i+n] = r[i][i];
    }
    printf("%d\n",ff());
    return 0;
}
```

181.碎纸机

```
#include<iostream>
#include<vector>

using namespace std;

const int factor[7] = {1, 10, 100, 1000, 10000, 100000, 1000000};
void dfs(vector<int> &ary, vector<int> &ans, int &max, bool &flag, int target, int sum, int num,
int bit)
{
    //剩余位数为 0，表示一次切割完成
    if(bit <= 0){
        //如果有两种切割方法都能得到最优的结果
        if(sum == max){
            flag = false;
        }
        else if(sum > max){
            flag = true;
            max = sum;
            ans = ary;
        }
    }
    else {
        for(int i = bit - 1; i >= 0; --i){
            if(sum + num / factor[i] <= target){
                ary.push_back(num/factor[i]);
                dfs(ary, ans, max, flag, target, sum + num / factor[i], num % factor[i], i);
                ary.pop_back();
            }
        }
    }
}

int main()
{
    int target;
    int num;
    cin >> target >> num;
    while(target != 0 || num != 0){
        //输入纸片上的数比目标数小
        if(num <= target){
            cout << num << ' ' << num << endl;
        }
        else {
            //bit 表示输入纸片上的数的位数
            int bit = 0;
            while(num >= factor[bit])
                bit++;
            vector<int> ary;//用于存放每一次搜索中的切割方法
            vector<int> ans;//用于存放当前为止最优路径的分割方法
```

```
                int max = -1;//表示当前为止最优路径的数字和
                bool flag = true;//表示当前为止是否有两种最优的切割方法
                dfs(ary, ans, max, flag, target, 0, num, bit);
                //无论怎么切纸片上数字和都大于目标数，没有一种合法的切割方法
                if(max < 0)
                        cout << "error" << endl;
                else {
                        if(flag){
                                cout << max;
                                for(vector<int>::iterator iter = ans.begin(); iter != ans.end(); ++iter)
                                        cout << ' ' << *iter;
                                cout << endl;
                        }
                        else
                                cout << "rejected" << endl;
                }
        }
        cin >> target >> num;
    }
    return 0;
}
```

182.文件结构图

```
#include<iostream>
#include<string>
#include<cstdio>
using namespace std;
void Sort(int N,string s[])
{
 int i=0;
 while(i<N-1)
 {
  int j=i+1;
  while(j<N)
  {
   if(s[i]>s[j])
   {
    string temp;
    temp=s[i];
    s[i]=s[j];
    s[j]=temp;
   }
   j++;
  }
  i++;
 }
}
char recur(string s1,int N,int T)
{
 string s2;
 cin>>s2;
 if(s2[0]=='#') return '#';
 if(s1=="ROOT") printf("DATA SET %d:\n",T);
 int i=0;
 while(i<N)
 {
  cout<<"|          ";
  i++;
 }
```

```
  cout<<s1<<endl;
  int j=0;
  string s[100];
  do
  {
   switch(s2[0])
   {
   case 'f':
     {
      s[j++]=s2;
     }
     break;
   case 'd':
     {
      recur(s2,N+1,T);
     }
     break;
   case ']':
     {
      Sort(j,s);
      int g=0;
      while(g<j)
      {
       int i=0;
       while(i<N)
       {
        cout<<"|          ";
        i++;
       }
       cout<<s[g]<<endl;
       g++;
      }
      return ']';
     }
   case '*':
     {
      Sort(j,s);
      int g=0;
      while(g<j)
      {
       cout<<s[g]<<endl;
       g++;
      }
      return '*';
     }
   case '#': return '#';
   }
  }while(cin>>s2);
  return '*';
}
int main()
{

  int T=1;
  while(true)
  {
   if(recur("ROOT",0,T)=='#') break;
   cout<<endl;
```

```
    T++;
  }
  return 0;
}
```

183.方格取数

可以发现在走到第 n 步时，能走到的格子是固定的。进一步发现，这些格子的横纵坐标加起来都是 n+1（左上角（1，1））。这样我们就不必同时记录两个坐标了，只要知道其中一个 t，另一个就可以通过 n+1-t 算出来。这个方法在一取方格（只走一条路）时效果并不明显，这里可以看到已经减少了一维，且越是高维优点越明显。用 f[x,i,j]表示走到第 x 步时，第 1 条路线走到横坐标为 i 的地方，第 2 条路线走到了横坐标为 j 的地方。这样，我们只要枚举 x，i，j，就能递推出来了。

```
#include <cstdio>
 #include <algorithm>
 using namespace std;
 int n;
 int f[31][11][11];
 int a[11][11];
 int main()
 {
        scanf("%d",&n);
        int x,y,c;
        while(scanf("%d%d%d",&x,&y,&c))
        {
              if (x == 0 && y == 0 && c == 0)
                    break;
              a[x][y] = c;
        }
        for (int k = 1; k < n + n; k ++)
              for (int i = 1; i <= min(n,k); i ++)
                    for (int j = 1; j <= min(n,k); j ++)
                    {
                          f[k][i][j] = max(f[k-1][i][j], f[k-1][i-1][j]);
                          f[k][i][j] = max(f[k][i][j], f[k-1][i][j-1]);
                          f[k][i][j] = max(f[k][i][j], f[k-1][i-1][j-1]);
                          if (i == j)
                                f[k][i][j] += a[i][k+1-i];
                          else       f[k][i][j] += a[i][k+1-i] + a[j][k+1-j];
                    }
        printf("%d\n",f[n+n-1][n][n]);
        return 0;
 }
```

184.过河

在河上有一座独木桥，一只青蛙想沿着独木桥从河的一侧跳到另一侧。在桥上有一些石子，青蛙很讨厌踩在这些石子上。

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
int a[105],f[20000],d[20000];
int main(){
       int l, m, s, t;
       memset(f,0,sizeof(f));
       memset(d,0,sizeof(d));
       cin >> l;
       cin >> s >> t >> m;
       for (int i = 1; i <= m; i ++){
             cin >> a[i];
```

```
    }
    int res = 0;
    if (s == t){
        for (int i = 1; i <= m; i ++){
            if (a[i] % s == 0){
                res ++;
            }
        }
        cout<<res<<endl;
    }
    else{
        sort(a+1, a+m+1);
        a[0] = 0;
        a[m+1] = l;
        for (int i = 0; i <= m; i ++){
            if (a[i+1] - a[i] > 90){
                a[i+1] = a[i] + (a[i+1] - a[i]) % 90;
            }
        }
        l = a[m+1];
        for (int i = 1; i <= m; i ++){
            d[a[i]] = 1;
        }
        for (int i = 1; i <= l + t; i ++){
            f[i] = 0xFFFFFFF;
        }
        for (int i = 1; i <= l + t; i ++){
            for (int j = s; j <= t; j ++){
                if (i >= j){
                    f[i] = min(f[i], f[i-j] + d[i]);
                }
            }
        }
        int minn = 0xFFFFFFF;
        for (int i = l; i <= l + t; i ++){
            if (f[i] < minn){
                minn = f[i];
            }
        }
        cout<<minn<<endl;
    }
}
```

185.装箱问题（dp）

有一个箱子容量为 V(正整数，0≤v≤20000)，同时有 n 个物品(o≤n≤30)，每个物品有一个体积 (正整数)。要求从 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。

```
#include<stdio.h>
int dp[31][20001]={0};
int num[31];
int max(int a,int b)
{
    return (a>b)?a:b;
}
int main(void)
{
    int i,j,k;
    int v,n;
    scanf("%d%d",&v,&n);
    for(i=1;i<=n;i++)
```

```
            scanf("%d",&num[i]);
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=v;j++)
            {
                if(i==1)
                {
                    if(j>num[i])
                        dp[i][j]=num[i];
                    else
                        dp[i][j]=0;
                }
                else
                {
                    if(j<num[i])
                        dp[i][j]=dp[i-1][j];
                    else
                        dp[i][j]=max(dp[i-1][j-num[i]]+num[i],dp[i-1][j]);
                }
            }
        }
        printf("%d",v-dp[n][v]);
        return 0;
}
```

186.poj1159Palindrome

```
#include<stdio.h>
char s[5001];
short dp[5001][5001];
short min(int a,int b)
{
    return (a<b)?a:b;
}
int main(void)
{
    short i,j,k;
    short n;
    scanf("%d",&n);
    scanf("%s",s);
    for(i=0;i<=n;i++)
        for(j=0;j<=n;j++)
            dp[i][j]=0;
    for(k=1;k<=n-1;k++)
    {
        for(j=0;j<n&&j+k<n;j++)
        {
            if(s[j]==s[j+k])
                dp[j][j+k]=dp[j+1][j+k-1];
            else if(s[j]!=s[j+k])
                dp[j][j+k]=min(dp[j][j+k-1],dp[j+1][j+k])+1;
        }
    }
    printf("%hd",dp[0][n-1]);
    return 0;
}
```

187. poj 1141 Brackets Sequence 括号匹配

```
#include <iostream>
#include <cstdio>
#include <cstring>
```

```
#include <cmath>
using namespace std;
const int maxn = 110;
char str[maxn];
int dp[maxn][maxn], path[maxn][maxn], len;
void output(int st, int ed);
int main()
{
	memset(dp, 0, sizeof(dp));
	scanf("%s", str + 1);
	len = strlen(str + 1);
	for(int i = 1; i <= len; i++)
		dp[i][i] = 1;
	for(int l = 2; l <= len; l++)
	{
		for(int i = 1; i <= len - l + 1; i++)
		{
			int j = i + l - 1;
			if(str[i] == '(' && str[j] == ')' || str[i] == '[' && str[j] == ']')
			{
				dp[i][j] = dp[i+1][j-1];
				path[i][j] = -1;
			}
			else
				dp[i][j] = 0x7fffffff;
			for(int k = i; k <= j - 1; k++)
			{
				if(dp[i][j] > dp[i][k] + dp[k+1][j])
				{
					dp[i][j] = dp[i][k] + dp[k+1][j];
					path[i][j] = k;
				}
			}
		}
	}
	output(1, len);
	printf("\n");
	return 0;
}
void output(int st, int ed)
{
	if(st > ed)
		return;
	else if(st == ed)
	{
		if(str[st] == '(' || str[ed] == ')')
			printf("()");
		else
			printf("[]");
	}
	else if(path[st][ed] == -1)
	{
		printf("%c", str[st]);
		output(st+1, ed - 1);
		printf("%c", str[ed]);
	}
	else
	{
```

```
            output(st, path[st][ed]);
            output(path[st][ed] + 1, ed);
        }
}
```

188. A decorative fence

```
#include <iostream>
using namespace std;
const int MAX_N = 25;
//木条的数目
int n;
//栅栏的编号
long long c;
//up[n][i]表示：n 个木条，且第一个木条长度为 i 的上升型栅栏数目
long long up[MAX_N][MAX_N];
//down[n][i]表示：n 个木条，且第一个木条长度为 i 的下降型栅栏数目
long long down[MAX_N][MAX_N];
bool used[MAX_N];
bool isPrint;
void FillTable()
{
    down[1][1] = 1;
    up[1][1] = 1;
    down[2][1] = 0;
    up[2][1] = 1;
    down[2][2] = 1;
    up[2][2] = 0;
    int i, j, k;
    for( i=3; i<MAX_N; i++ )
    {
        //??? j<=MAX_N
        for( j=1; j<MAX_N; j++ )
        {
            //??? k=j
            for( k=j; k<i; k++ )
            {
                up[i][j] += down[i-1][k];
            }
            for( k=1; k<j; k++ )
            {
                down[i][j] += up[i-1][k];
            }
        }
    }
}
void SolveAndPrint( int id, int count )
{
    for( int i=1; i<=count; i++ )
    {
        if( used[i] && i<=id )
        {
            id++;
            count++;
        }
    }
    used[id] = true;
    if( isPrint )
    {
        cout << " " << id;
```

```
        }
        else
        {
                cout << id;
                isPrint = true;
        }
}
int main()
{
        //freopen( "in.txt", "r", stdin );
        int caseNum;
        cin >> caseNum;
        //DP 的填表
        FillTable();
        while( caseNum-- )
        {
                cin >> n >> c;
                //先搜索 down
                bool isDown = true;
                bool isFirst = true;
                int id = 1;
                isPrint = false;
                memset( used, 0, sizeof(used) );
                while( n )
                {
                        if( isDown )
                        {
                                if( c > down[n][id] )
                                {
                                        c -= down[n][id++];

                                        if( isFirst )
                                        {
                                                //设置标记，下次搜索 up
                                                isDown = false;
                                                //对每个 id 有 down 和 up 两个值
                                                //确定第一个数时，down 和 up 都要搜索
                                                id--;
                                        }
                                }
                                else
                                {
                                        //如果当前木条长度可确定
                                        //问题规模减一
                                        SolveAndPrint( id, n-- );
                                        isFirst = false;
                                        isDown = false;
                                        id = 1;
                                }
                        }
                        else
                        {
                                if( c > up[n][id] )
                                {
                                        c -= up[n][id++];
                                        if( isFirst )       isDown = true;
                                }
                                else
```

```
                {
                    SolveAndPrint( id, n-- );
                    isFirst = false;
                    isDown = true;
                }
            }
        }
        cout << endl;
    }
    return 0;
}
```

189. The Cow Lexicon 奶牛词典

题意就是给出一个主串，和一本字典，问最少在主串删除多少字母，可以使其匹配到字典的单词序列。

**dp[i]表示从 message 中第 i 个字符开始，到第 L 个字符（结尾处）这段区间所删除的字符数，初始化为 dp[L]=0**

**由于我的程序是从 message 尾部向头部检索匹配，所以是下面的状态方程：**

$$\begin{cases} dp[i]=dp[i+1]+1 & \text{不能匹配时（最坏情况，删除最多字符）} \\ dp[i]=\min\{dp[i],dp[pm]+(pm-i)-len\} & \text{（可以匹配时，取最优）} \end{cases}$$

```
#include<iostream>
#include<string>
using namespace std;
int min(int a,int b)
{
    return a<b?a:b;
}
int main(int i,int j)
{
    int w,L;
    while(cin>>w>>L)
    {
        /*Read In*/
        int* dp=new int[L+1];
        char* mesg=new char[L];
        string* dict=new string[w];
        cin>>mesg;
        for(i=0;i<w;i++)
            cin>>dict[i];
        /*Initial*/
        dp[L]=0;            //dp[i]表示从 i 到 L 所删除的字符数
        /*Dp-Enum*/
        for(i=L-1;i>=0;i--)   //从 message 尾部开始向前检索
        {
            dp[i]=dp[i+1]+1;   //字典单词和 message 无法匹配时，删除的字符数（最坏的情况）
            for(j=0;j<w;j++) //对字典单词枚举
            {
```

```
                int len=dict[j].length();
                if(len<=L-i && dict[j][0]==mesg[i])   //单词长度小于等于当前待匹配
message 长度
                {                                                    //且单词头字母与信息第 i 个
字母相同
                    int pm=i;   //message 的指针
                    int pd=0;   //单词的指针
                    while(pm<L) //单词逐字匹配
                    {
                        if(dict[j][pd]==mesg[pm++])
                            pd++;
                        if(pd==len)
                        {       //字典单词和 message 可以匹配时，状态优化（更新）
                            dp[i]=min(dp[i],dp[pm]+(pm-i)-len);//dp[pm]表示从 pm 到 L
删除的字符数
                            break;                                   //(pm-i)-pd 表示从 i
到 pm 删除的字符数
                        }                                            //则
dp[pm]+(pm-i)-pd 表示从 i 到 L 删除的字符数
                    }
                }
            }
        }
        cout<<dp[0]<<endl;

        delete dp,mesg,dict;
    }
    return 0;
}
```

190.dividing 邮票分配

有分别价值为 1,2,3,4,5,6 的 6 种物品，输入 6 个数字，表示相应价值的物品的数量，问一下能不能将物品分成两份，是两份的总价值相等，其中一个物品不能切开，只能分给其中的某一方，当输入六个 0 是（即没有物品了），这程序结束

```
#include<iostream>
using namespace std;
int n[7];   //价值为 i 的物品的个数
int SumValue;   //物品总价值
int HalfValue;   //物品平分价值
bool flag;      //标记是否能平分 SumValue
void DFS(int value,int pre)
{
    if(flag)
        return;
    if(value==HalfValue)
    {
        flag=true;
        return;
    }
    for(int i=pre;i>=1;i--)
    {
        if(n[i])
        {
            if(value+i<=HalfValue)
            {
                n[i]--;
                DFS(value+i,i);

                if(flag)
```

```
                        break;
                }
            }
        }
        return;
}
int main(int i)
{
        int test=1;
        while(cin>>n[1]>>n[2]>>n[3]>>n[4]>>n[5]>>n[6])
        {
            SumValue=0;   //物品总价值
            for(i=1;i<=6;i++)
                SumValue+=i*n[i];
            if(SumValue==0)
                break;
            if(SumValue%2)       //sum 为奇数，无法平分
            {
                cout<<"Collection #"<<test++<<':'<<endl;
                cout<<"Can't be divided."<<endl<<endl;       //注意有空行
                continue;
            }
            HalfValue=SumValue/2;
            flag=false;
            DFS(0,6);
            if(flag)
            {
                cout<<"Collection #"<<test++<<':'<<endl;
                cout<<"Can be divided."<<endl<<endl;
                continue;
            }
            else
            {
                cout<<"Collection #"<<test++<<':'<<endl;
                cout<<"Can't be divided."<<endl<<endl;
                continue;
            }
        }
        return 0;
}
```

191.AGTC

dp[i][j]，代表字符串 1 的前 i 子串和字符串 2 的前 j 子串的距离。初始化 dp[0][0] = 0,dp[i][0] = i,dp[0][j] = j;

dp[i][j] = min {dp[i][j-1] + 1//添加(在 i 位置后面添加一个字符)dp[i-1][j] + 1//删除(删除第 i 个字符)dp[i-1][j-1] + (A[i] == B[i]?0:1)       //替换              }

```
#include<stdio.h>
char A[1001];
char B[1001];
int dp[1001][1001];
int min(int a,int b,int c)
{
        if(a>=b)
        {
            if(c>=b)
                return b;
            return c;
        }
        else
```

```
        {
            if(c>=a)
                return a;
            return c;
        }
}
int main(void)
{
        int i,j,k;
        int n,m;
        while(scanf("%d",&n)==1)
        {
            scanf("%s",A);
        scanf("%d",&m);
        scanf("%s",B);
        for(i=0;i<=1000;i++)
            for(j=0;j<=1000;j++)
                dp[i][j]=100000000;
        for(i=0;i<=n;i++)
            dp[i][0]=i;
        for(i=0;i<=m;i++)
            dp[0][i]=i;
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=m;j++)
            {
                dp[i][j]=min(dp[i][j-1]+1,dp[i-1][j]+1,dp[i-1][j-1]+(A[i-1]==B[j-1]?0:1));
            }
        }
        printf("%d\n",dp[n][m]);
        }
        return 0;
}
```

192.生日相同

```
#include<stdio.h>
#define N 100
int main ()
{
        int a[N][2], n, m=0, h=0, i, j, k;
        char s[N][12];
        scanf ( "%d", &n );
        for ( i=0; i<n; i++ )
            scanf ( "%s %d %d", s[i], &a[i][0], &a[i][1] );
        for ( i=1; i<=12; i++ )
        {
            m=0;
            for ( j=0; j<n; j++ )
                if ( a[j][0] == i )
                    m++;
            if ( m>1 )
            {
        for ( k=1; k<32; k++ )
        {
        h=0;
        for ( j=0; j<n; j++ )
        if ( a[j][0] == i && a[j][1] == k )
        h++;
        if ( h>1 )
```

```
        {
            printf ( "%d %d", i, k );
              for ( j=0; j<n; j++ )
              if ( a[j][0] == i && a[j][1] == k )
              printf ( " %s", s[j] );
                printf ( "\n" );
        }
        }
        }
        }
        return 0;
}
```

193.正整数的任意进制转换（大整数）

```
#include <stdio.h>
#include <string.h>
char str1[500], str2[500], quto[500];   //str1 保存输入，str2 保存输出，quto 保存商
int char_2_int(char c){   //字符映射到整数
        int n;
        if(c >= '0' && c <= '9') n = c-'0';
        else if(c >= 'A' && c <= 'Z') n = c-'A'+10;
        else n = c-'a'+36;
        return n;
}
char int_2_char(int n){   //整数映射到字符
        char c;
        if(n >= 0 && n <= 9) c = n+'0';
        else if(n >= 10 && n <= 35) c = n+'A'-10;
        else c = n+'a'-36;
        return c;
}
int main(){
        int b1, b2, n, s, cur, last, top;   //b1 为转换前进制，b2 为转换后进制
        int t, quit;
        int i, j, k;
        char c;
        scanf("%d", &t);                //有几组数据
        for(k = 0; k < t; k++){
                scanf("%d,",&b1);
                i=0;
                while((c=getchar())!=',')
                        str1[i++]=c;
                str1[i]='\0';
                scanf("%d",&b2);
                //scanf("%d,%s,%s", &b1, &b2, str1);
                //printf("%d %s\n%d ", b1, str1, b2);
                top = 0;
                quit = 0;
                while(!quit){
                        last = 0;
                                        for(i = 0; i < strlen(str1); i++){   //一次求商取余（关键）
                                                cur = last*b1+char_2_int(str1[i]);   //对输入数从高位开
始进行逐位加权累加，存入 cur
                                if(cur < b2){   //若 cur 的值小于基 b2，则继续累加，且记商的对应位为 0
                                        last = cur;
                                        quto[i] = '0';
                                        continue;
                                }
```

```
                    else{   //若 cur 的值大于基 b2，则进行除法运算，保存商的对应位，本次余数纳入加权累加中
                        quto[i] = int_2_char(cur/b2);
                        last = cur%b2;
                    }
                }
                quto[i] = '\0';
                quit = 1;
                for(j = 0; j < strlen(quto); j++)   //检测商是否为 0，若为 0 则表示转换完毕，结束循环
                    if(quto[j] != '0') quit = 0;
                str2[top++] = int_2_char(last);   //保存本次除法运算的余数作为转换后新数的一位
                for(s = 0; s < strlen(quto); s++)
                    if(quto[s] != '0') break;
                for(i = s; i < strlen(quto); i++)   //将本次除法运算的商作为新的被除数
                    str1[i-s] = quto[i];
                str1[i-s] = '\0';
            }
            for(int i = top-1; i >= 0; i--)   //逐位逆推得转换后的新数
                printf("%c", str2[i]);
            printf("\n");
        }
        return 0;
}
```

194.最长等差数列子集

```
#include<stdio.h>
#include<stdlib.h>
int dp[5000][5000];         //dp[i][j]表示以第 i 个数开头公差为 j 的最长子序列长度
int num[5000];
int comp(const void *a,const void *b)
{
    return *(int *)a-*(int *)b;
}
int max(int a,int b)
{
    return (a>b)?a:b;
}
int main(void)
{
    int i,j,k;
    int n;
    scanf("%d",&n);
    for(i=0;i<n;i++)
        scanf("%d",&num[i]);
    qsort(num,n,sizeof(num[0]),comp);
    int maxlength=0,maxstart=0,maxcha=0;
    for(i=n-2;i>=0;i--)                    //从后向前
    {
        for(j=i+1;j<=n-1;j++)
        {
            dp[i][num[j]-num[i]]=max(dp[i][num[j]-num[i]],max(dp[j][num[j]-num[i]],1)+1);
            if(maxlength<dp[i][num[j]-num[i]])
            {
                maxlength=dp[i][num[j]-num[i]];
                maxstart=i;
                maxcha=num[j]-num[i];
            }
```

```
                else if(maxlength==dp[i][num[j]-num[i]])
                {
                    if(num[j]-num[i]>maxcha)
                    {
                        maxcha=num[j]-num[i];
                        maxstart=i;
                    }
                    else if(num[j]-num[i]==maxcha)
                    {
                        if(i>maxstart)
                            maxstart=i;
                    }
                }
            }
        }
        if(maxlength<=2)
            printf("NO");
        else
        {
            int x=num[maxstart];
            for(i=1;i<maxlength;i++,x+=maxcha)
                printf("%d,",x);
            printf("%d",x);
        }
        return 0;
}195.POJ2192 Zipper
若用 DP 来作先定义 res[i][j]=1 表示串 1 前 i 个字符和串 2 的前 j 个字符能组成串 3 的前 i+j 个字符，res[i][j]=0 则不能。态转移方程如下：
Res[i][j]= (third[i+j]==first[i] && res[i-1][j]==1) ||(third[i+j]==second[j]&&res[i][j-1]==1)
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
char first[201],second[201],third[401];
int res[201][201];
int init(int n,int m)
{
    int i;
    for(i=1;i<=m;i++)
        if(second[i]==third[i]) res[0][i]=1;
        else break;
    for(i=1;i<=n;i++)
        if(first[i]==third[i]) res[i][0]=1;
        else break;
    return 0;
}
int dp(int n,int m)
{
    int i,j;
    for(i=1;i<=n;i++)
        for(j=1;j<=m;j++)
        {
            if(third[i+j]==first[i] && res[i-1][j]) res[i][j]=1;
            if(third[i+j]==second[j] && res[i][j-1]) res[i][j]=1;
        }
    if(res[n][m]) return 1;
    return 0;
}
```

```
int main()
{
	int n,len1,len2,count=0;;
	scanf("%d",&n);
	while(n--)
	{
		count++;
		scanf("%s %s %s",first+1,second+1,third+1);
		len1=strlen(first+1);
		len2=strlen(second+1);
		memset(res,0,sizeof(res));
		init(len1,len2);

		if(dp(len1,len2))
			printf("Data set %d: yes\n",count);
		else
			printf("Data set %d: no\n",count);
	}
	return 0;
}
```

196.找次大数

```
#include<stdio.h>
#include<string.h>
int num[10000]={0};
char s[3000];
int main(void)
{
	int i,j=0,k=0;
	char c;
	scanf("%s",s);
	for(i=0;i<strlen(s);i++)
	{
		if(s[i]>='0'&&s[i]<='9')
			k=k*10+s[i]-'0';
		else
		{
			num[k]++;
			k=0;
		}
		if(i==strlen(s)-1&&k!=0)
			num[k]++;
	}
	int ok=0;
	for(i=9999;i>=0;i--)
	{
		if(num[i]>0&&j==0)
			j=1;
		else if(num[i]>0&&j==1)
		{
			printf("%d",i);
			ok=1;
			break;
		}
	}
	if(ok==0)
		printf("No");
	return 0;
}
```

197.两倍

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
	int num[16]={0},i=0,j=0,sum=0;
	while(1)
	{
		scanf("%d",&num[0]);
		if(num[0]==-1) break;
		for(i=1;num[i-1]!=0;i++)
		{
			scanf("%d",&num[i]);
		}
		for(i=0;num[i]!=0;i++)
		{
			for(j=0;num[j]!=0;j++)
			{
				if((num[i]==2*num[j])&&(num[i]!=0)&&(num[j]!=0))
				sum++;
			}
		}
		cout<<sum<<endl;
		sum=0;
	}
	return 0;
}
```

198.循环移动

```
#include<stdio.h>
int main()
{
	int n,m;
	scanf("%d%d",&n,&m);
	int a[200];
	int i;
	for(i=0;i<n;i++)
		scanf("%d",&a[100+i]);
	for(i=0;i<m;i++)
		a[100-m+i]=a[100+n-m+i];
	for(i=100-m;i<99+n-m;i++)
		printf("%d ",a[i]);
	printf("%d",a[99+n-m]);
}
```

199.字符串排序

```
#include<stdio.h>
#include<string.h>
void paixu(int n,char a[100][200])
{
	int i,j;
	char ch[200];
	for(i=0;i<n-1;i++)
	{
		for(j=0;j<n-i-1;j++)
		{
			if(strcmp(a[j],a[j+1])>0)
			{
				strcpy(ch,a[j]);
```

```
                    strcpy(a[j],a[j+1]);
                    strcpy(a[j+1],ch);
                }
            }
        }
}
int main()
{
    int n,t;
    char a[100][200];
    int i,j,k;
    scanf("%d\n",&t);
    for(i=0;i<t;i++)
    {
        scanf("%d\n",&n);
        for(j=0;j<n;j++)
        {
            gets(a[j]);
        }
        paixu(n,a);
        for(k=0;k<n;k++)
        {
                printf("%s\n",a[k]);
        }
        printf("\n");
    }
}
```

200.DNA 排序

```
#include<iostream>
#include<algorithm>
#define INF 2147483647
using namespace std;
struct str
{
    char source[100];
    char s[100];
    int misorder;
}DNA[101];
int len,n,_min,flag,cnt;
char T[100];
int merge_sort(char A[],int x,int y,char T[])
{
    if(y-x > 1)
    {
        int m = (x+y) / 2;
        int p = x,q = m,i = x;
        merge_sort(A,x,m,T);
        merge_sort(A,m,y,T);
        while(p < m || q < y)
        {
            if(q >= y || (p < m && A[p] <= A[q]))
                T[i++] = A[p++];
            else
            {
                T[i++] = A[q++];
                cnt += m - p;
            }
        }
```

```
            for(i = x;i < y;++i)
                A[i] = T[i];
        }
        return cnt;
}
int main()
{
        scanf("%d%d",&len,&n);

        for(int i = 0;i < n;++i)
        {
            scanf("%s",DNA[i].s);
            strcpy(DNA[i].source,DNA[i].s);
            cnt = 0;
            DNA[i].misorder = merge_sort(DNA[i].s,0,len,T);
        }
        for(int j = 0;j < n;++j)
        {
            _min = INF;
            for(int i = 0;i < n;++i)
            {
                if(DNA[i].misorder < _min)
                {
                    _min = DNA[i].misorder;
                    flag = i;
                }
            }
            printf("%s\n",DNA[flag].source);
            DNA[flag].misorder = INF;
        }
        return 0;
}
```

201.和为 n 连续正数序列

```
#include<stdio.h>
int main()
{
        int n;
        scanf("%d",&n);
        int start,end,i;
        int count=0;
        for(start=1;start<n/2+1;start++)
        {
            for(end=start+1;end<=n/2+1;end++)
            {
                if(n==(end+start)*(end-start+1)/2)
                {
                    for(i=start;i<end;i++)
                        printf("%d ",i);
                    printf("%d\n",end);
                    count++;
                }
            }
        }
        if(count==0)
            printf("NO");
}
```

202.最长最短单词

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

```
#include <string.h>
int main()
{
	char s[1000];
	int x,y,m,n,i,j;
	i=0;
	j=0;
	x=0;
	y=20;
	m=0;
	n=0;
	gets(s);
	while(s[i]!='\0')
	{
		while(s[i]!=' '&&s[i]!=','&&s[i]!='\0')
		{
			j++;
			i++;
		}
		if(j>x)
		{
			m=i-j;
			x=j;
		}
		if(j<y)
		{
			n=i-j;
			y=j;
		}
		while(s[i]==' '||s[i]==',')
			i++;
		j=0;
	}
	while(s[m]!=' '&&s[m]!=','&&s[m]!='\0')
	{
		printf("%c",s[m]);
		m++;
	}
	printf("\n");
	while(s[n]!=' '&&s[n]!=','&&s[n]!='\0')
	{
		printf("%c",s[n]);
		n++;
	}
	printf("\n");
	return 0;
}
```

203.提取数字串按数值排序

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int comp(const void *a,const void *b)
{
	return *(int *)a-*(int *)b;
}
int num[1000];
char s[1000];
int main(void)
```

```
{
	//freopen("in.txt","r",stdin);
	int i,j=0,k=0,ok,flag=0;
	scanf("%s",&s);
	for(i=0;i<strlen(s);i++)
	{
		if(s[i]>='0'&&s[i]<='9')
			k=k*10+s[i]-'0';
		else if(s[i-1]>='0'&&s[i-1]<='9')
		{
			num[j++]=k;
			k=0;
		}
		if(i==strlen(s)-1&&k!=0)
		{
			num[j]=k;
			flag=1;
		}
	}
	if(flag==1)
	{
		qsort(num,j+1,sizeof(num[0]),comp);
		for(i=0;i<j;i++)
		printf("%d,",num[i]);
		printf("%d",num[i]);
	}
	else
	{
		qsort(num,j,sizeof(num[0]),comp);
		for(i=0;i<j-1;i++)
		printf("%d,",num[i]);
		printf("%d",num[i]);
	}
	return 0;
}
```

204.降序生成进制数

```
#include <stdio.h>
void f(char a[],long long int i,int k,int n)
{
	long long int j;
	for(j=n-1;j>=0;j--)
	{
		a[j]=i%k+48;
		i=i/k;
	}
}
int main()
{
	long int m,n,k;
	long long int i,j,t=1;
	scanf("%d %d %d",&m,&n,&k);
	char a[100]={0};
	for(i=1;i<=n;i++)
		t=t*k;
	if(m>t)
		m=t;
	for(i=t-1;i>=t-m;i--)
	{
```

```
            f(a,i,k,n);
            printf("%s",a);
            if((t-i)%6==0||i==t-m)
                  printf("\n");
            else printf(",");
      }
}
```

205.木棒

```
#include "iostream"
#include "cstdio"
#include "cstring"
#include "algorithm"
#include "functional"
using namespace std;
#define maxn 65
int n, sum, goal;
int stick[maxn];
bool visit[maxn];
bool cmp(const int &a, const int &b)
{
    return a > b;
}

bool dfs(int now, int index, int cnt) {
      if(goal * cnt == sum) return true;
      for(int i = index; i < n; i++) {
            if(visit[i] || (i && !visit[i-1] && stick[i] == stick[i-1])) continue;
            if(now + stick[i] == goal) {
                  visit[i] = true;
                  if(dfs(0, 0, cnt + 1)) return true;
                  visit[i] = false;
                  return false;
           } else if(now + stick[i] < goal) {
                              visit[i] = true;
              if(dfs(now + stick[i], i + 1, cnt)) return true;
             visit[i] = false;
              if(now == 0) return false;
            }       }
       return false;
 }
 int solve() {
       sort(stick, stick + n, cmp);
       for(goal = stick[0]; goal < sum; goal++) {
             if(sum % goal != 0) continue;
            memset(visit, false, sizeof(visit));
            if(dfs(0, 0, 0)) break;
       }
      return goal;
 }

 int main() {
      while(~scanf("%d", &n)) {
            if(!n) break;
           sum = 0;
            for(int i = 0; i < n; i++) {                    scanf("%d", &stick[i]);
                  sum += stick[i];
            }
            printf("%d\n", solve());
```

```
        }
        return 0;
  }
```
206.字符串重排列
```
#include<stdio.h>
int main()
{
    int i,j,c[200]={0},d[200]={0};
    char a[100],b[100];
    scanf("%s",a);
    scanf("%s",b);
    for(i=0;a[i]!='\0';i++)
    {
        j=a[i]-'0';
        c[j]++;
    }
    for(i=0;b[i]!='\0';i++)
    {
        j=b[i]-'0';
        d[j]++;
    }
    for(i=0;i<200;i++)
    {
        if(c[i]!=d[i])
        {
            printf("NO");
            goto Loop;
        }
    }
    printf("YES");
Loop:   printf("\n");
    return 0;
}
```
207.计算矩阵边缘元素之和
```
#include<iostream>
#include<cstring>
#include<stdio.h>
#include <cmath>            //预编译命令
#include<iomanip>
using namespace std;        //使用名字空间
int main ()
{
  int k = 0,m ,n , i ,j ,r,s;
  cin >> k;//输入 k 组数据；
  for( s = 0; s < k; s++)
  {
    int sum = 0;//定义和为 0；
    cin >> m;//输入行数
    cin >> n;//输入列数
    int a[10000] = {0};//清空数组
    for( i = 0; i < m * n ; i++)//输入矩阵
    {
      cin >> a[i];
    }
    int *p;
    p = a;
    for(i =0; i <= n; i++)
    {
```

```
    sum += *(p + i);
   }
   for(j =    2* n - 1; j <    (m - 1) * n; j = j + n -1)
   {
    sum += *( p+j);
    j++;
    sum += *(p+j);
   }
   for(r = (m-1) * n +1 ; r <    m * n; r++)
   {
    sum += * (r + p);
   }
   cout << sum << endl;
  }
  return 0;
}
```

208.中间值判断

```
#include <stdio.h>
#include <string.h>
#include <math.h>
int main()
{
	int a[100],min,max,i,n,temp;
	scanf("%d\n",&n);
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);
	max=a[0];
	min=a[0];
	for(i=0;i<n;i++)
	{
		if(max<=a[i])
			max=a[i];
	}
	for(i=0;i<n;i++)
	{
		if(min>=a[i])
			min=a[i];
	}
	if((max+min)%2==0)
	{
		for(i=0;i<n;i++)
		{
			if(a[i]==(max+min)/2)
				temp=1;
		}
	}
	if((max+min)%2==1)
	{
		for(i=0;i<n;i++)
		{
			if(a[i]==(max+min)/2||a[i]==(max+min)/2+1)
				temp=1;
		}
	}
	if(temp==1)
		printf("YES");
	else
		printf("NO");
```

```
}
```
209. 判断整数的奇偶性
```
#include <stdio.h>
#include <string.h>
int main()
{
	char a[600],b[600],c[600];
	int i,j,r=0,s=0,l1,l2;
	gets(a);
	l1=strlen(a);
	for(i=0;i<l1;i++)
	{
		if(a[i]>='0'&&a[i]<='9')
		{
			b[r]=a[i];
			r++;
		}
	}
	b[r]='\0';
	l2=strlen(b);
	j=0;
	while(j<l2)
	{
		if(b[j]=='0')
			j++;
		else
			break;
	}
	for(i=j;b[i]!='\0';i++)
	{
		c[s]=b[i];
		s++;
	}
	c[s]='\0';
	if((c[s-1]-'0')%2==0)
		printf("%s,EVEN",c);
	else if(b[0]!='\0'&&c[0]=='\0')
		printf("0,EVEN");
	else if(b[0]=='\0')
		printf("NO");
	else
		printf("%s,ODD",c);
}
```
210.最简真分数序列
```
#include<stdio.h>
#include<stdlib.h>
int comp(const void *a,const void *b)
{
	return *(int *)a-*(int *)b;
}
int num[201];
int f(int i,int j)
{
	int k;
	if(i>=j||i==0)
		return 0;
	for(k=2;k<=i;k++)
	{
```

```
            if(i%k==0&&j%k==0)
                return 0;
        }
        return 1;
}
int main(void)
{
        int i,j,k,ok=0;
        int n;
        scanf("%d",&n);
        num[0]=0;
        for(i=1;i<=n;i++)
            scanf("%d",&num[i]);
        qsort(num,n+1,sizeof(num[0]),comp);
        for(i=1;i<n;i++)
            for(j=i+1;j<=n;j++)
            {
                if(f(num[i],num[j])&&ok==0)
                {
                    printf("%d/%d",num[i],num[j]);
                    ok=1;
                }
                else if(f(num[i],num[j])&&ok==1)
                    printf(",%d/%d",num[i],num[j]);
            }
        if(ok==0)
            printf("NO");
        return 0;
}
```

211.1102 迷宫

```
#include<stdio.h>
int n,ok;
int x1,y1;
char s[101][101];
int dp[101][101];
void f(int i,int j)
{
        dp[i][j]=1;
        if(s[i][j]=='#')
            ;
        else if(i==x1&&j==y1)
            ok=1;
        else
        {
            if(i+1<n&&s[i+1][j]!='#'&&dp[i+1][j]==0)
                f(i+1,j);
            if(i-1>=0&&s[i-1][j]!='#'&&dp[i-1][j]==0)
                f(i-1,j);
            if(j+1<n&&s[i][j+1]!='#'&&dp[i][j+1]==0)
                f(i,j+1);
            if(j-1>=0&&s[i][j-1]!='#'&&dp[i][j-1]==0)
                f(i,j-1);
        }
}
int main(void)
{
        //freopen("in.txt","r",stdin);
        int i,j,k,x,y;
```

```
    scanf("%d",&k);
    while(k--)
    {
        scanf("%d",&n);
        for(i=0;i<n;i++)
            scanf("%s",s[i]);
        ok=0;
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                dp[i][j]=0;
        scanf("%d%d%d%d",&x,&y,&x1,&y1);
        f(x,y);
        if(ok==1)
            printf("YES\n");
        else
            printf("NO\n");
    }
    return 0;
}
```

212.字符串最大跨距

```
#include<stdio.h>
#include<string.h>
char s[301];
char s1[11];
char s2[11];
int main(void)
{
    int i=0,j,k,left=-1,right=-1;
    char c;
    while((c=getchar())!=',')
        s[i++]=c;
    s[i]='\0';
    i=0;
    while((c=getchar())!=',')
        s1[i++]=c;
    s1[i]='\0';
    scanf("%s",s2);
    for(i=0;i<strlen(s);i++)
    {
        for(k=i,j=0;j<strlen(s1);k++,j++)
        {
            if(s[k]!=s1[j])
                break;
        }
        if(j==strlen(s1))
        {
            left=k-1;
            break;
        }
    }
    for(i=0;i<strlen(s);i++)
    {
        for(k=i,j=0;j<strlen(s2)&&k<strlen(s);k++,j++)
        {
            if(s[k]!=s2[j])
                break;
        }
        if(j==strlen(s2))
```

```
            {
                right=i;
            }
        }
        if(left<right&&left>=0&&right>=0)
            printf("%d",right-left-1);
        else if(left>=0&&right>=0)
            printf("-1");
        else
            printf("-1");
        return 0;
}
```

213.啤酒厂选址

```
#include<stdio.h>
int num[10000];          //记录每个居民点需要啤酒桶数
int a[10000];            //记录距离
int mindis[10000][10000];      //记录两个点之间的最短距离
int dis[10000][10000];      //记录两个点之间的顺时针距离
int min(int c,int d)
{
    return (c>d)?d:c;
}
int main(void)
{
    int i,j,k,sum=0;
    int n;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d%d",&num[i],&a[i]);
        sum+=a[i];
        dis[i][i+1]=a[i];
        dis[i+1][i]=a[i];
    }
    for(i=0;i<n;i++)
        for(j=i+2;j<n;j++)
            dis[i][j]=dis[i][j-1]+a[j-1];
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            mindis[i][j]=min(dis[i][j],sum-dis[i][j]);
            mindis[j][i]=mindis[i][j];
        }
    }
    int minmin=999999999;
    int t=0;
    int start;
    for(i=0;i<n;i++)
    {
        t=0;
        for(j=0;j<n;j++)
            t+=mindis[i][j]*num[j];
        if(minmin>t)
        {
            minmin=t;
            start=i;
        }
```

```
        }
        printf("%d,%d",start,minmin);
        return 0;
}
```

214.柱状图上的最大矩形

```
#include<stdio.h>
int dp[20001];
int f(int n);
int main(void)
{
        int i,j,k;
        int n;
        scanf("%d",&n);
        for(i=1;i<=n;i++)
                scanf("%d",&dp[i]);
        printf("%d",f(n));
        return 0;
}
int f(int n)
{
        int j,k,l,t;
        int max=0;
        for(j=1;j<=n;j++)
        {
                if(dp[j]>0)
                {
                        t=1;
                        for(k=j+1;k<=n;k++)
                        {
                                if(dp[k]>=dp[j])
                                        t++;
                                else
                                        break;
                        }
                        for(k=j-1;k>0;k--)
                        {
                                if(dp[k]>=dp[j])
                                        t++;
                                else
                                        break;
                        }
                        if(t*dp[j]>max)
                                max=t*dp[j];
                }
        }
        return max;
}
```

215.最大零矩阵

```
#include<stdio.h>
int n,m;
int num[101][101];    //用来储存每个数字
int dp[101][101];     //用来储存到从第 i 行开始第 j 行为止每一列的和
int f(int j)              //求一维数组的最大子序列和
{
        int i,max=0,sum=0;
        for(i=1;i<=n;i++)
        {
                sum+=dp[j][i];
```

```
            if(sum>max)
                max=sum;
            else if(sum<0)
                sum=0;
        }
        return max;
}
int main(void)
{
        freopen("in.txt","r",stdin);
        int i,j,k,x,y,z;
        int max=-100000;
        scanf("%d%d",&m,&n);
        for(i=1;i<=m;i++)
            for(j=1;j<=n;j++)
            {
                scanf("%d",&k);
                if(k==0)
                    num[i][j]=1;
                else
                    num[i][j]=-9999;
            }
        for(i=1;i<=m;i++)                       //从第 i 行开始
        {
            for(j=i;j<=m;j++)                   //到第 j 行
            {
                if(i==j)
                {
                    for(k=1;k<=n;k++)
                        dp[j][k]=num[j][k];
                    x=f(j);
                    if(x>max)
                        max=x;
                }
                else
                {
                    for(k=1;k<=n;k++)
                        dp[j][k]=dp[j-1][k]+num[j][k];
                    x=f(j);
                    if(x>max)
                        max=x;
                }
            }
            for(y=1;y<=m;y++)               //清零数组
                for(z=1;z<=n;z++)
                    dp[y][z]=0;
        }
        printf("%d",max);
        return 0;
}
```