练习题

一 动态规划题

1．兔子繁殖问题（经典的斐波那契数列）

假设一对新生兔子，从第三个月开始，每个月都会下t=1对新生兔子，新生的兔子具有同样规律，而且假定兔子不死亡，现在从第一个月1对兔子开始，求第n个月最末一天兔子有多少对?

除了t=1的情况，你能求出t=2的数列结果吗？（显然前几项依次为1,1,3,5,11,….）

假设兔子只养6个月，即在第6个月其下完小兔子后会被卖出，再求第n个月最后一天兔子总数。//值得仔细思考

2．神奇植物（斐波那契数列升级版）

（时间限制：5000 ms | 内存限制：16384 KB ）

前不久，探险队b前往inst沙漠进行考察。他们在沙漠中发现了一种生命力十分强悍的植物。这种植物从种子到发芽只需要1天时间，而从发芽的第x天开始，一直到第y天（包括第y天），每天又都会繁殖出一粒种子，每棵植物都会在发芽的第z天时枯萎死亡。

因为探险队b以前在营地里并没有见过这种植物，于是他们取了一颗种子放在他们的营地里（这一天叫做第0天），他们想知道第n天的时候营地里这种神奇的植物(不包括种子)的棵数m。

但是这个数太大了，他们现在只想知道m的最后8位数是多少。

输入

题目包含多个测试数据，每个测试数据只有一行，包含四个正整数x，y，z，n，用空格分开。其中0 < x <= y < z <= 2500，0 < n <= 2500。题目以四个整数0 0 0 0结束输入，这一行不用处理。

输出

每个测试数据输出一行，包含一个正整数，表示m的最后8位，m不足八位的输出m。输出不含前导0。

样例输入

1 2 3 4
0 0 0 0

样例输出

5

//此题与上面兔子繁殖问题中最后一问是一样的。

3．最少硬币问题

有一组面值分别为a1,a2,a3,…,ak（假设已经递增排列）的硬币，每种硬币数量足够多，现在要从这些硬币中抽取最少的枚数组成总金额n，请设计算法求出这个最少枚数。

输入

题目包含多组测试数据，每一组测试数据由2行构成，第一行为整数k和n，紧接着一行会有k个从小到大顺序的整数，最后一行以0 0结束。

输出

每一组测试数据单独一行输出其对应最少硬币枚数。

样例输入

```
4 10
1 2 3 4
4 8
1 2 4 5
0 0
```

样例输出

```
3
2
```

4．平方数

众所周知，一个正整数可以分成若干个正整数的和（废话），现在我们把问题改一下。把一个正整数分成若干个平方数的和，显然这也是一定可以达到的。例如8 = 4 + 4 = 2^2 + 2^2 41 = 4^2 + 5^2 3 = 1 + 1 + 1。但是现在试卷空格有限，不可能让你写n个1上去，所以我们需要最少的分拆。

Input

本题包含多组测试数据，每行是一个正整数n(1<=n<=100000)。

Output

对于每个数据，需要输出两行，第一行为最少的平方数的数量，第二行是这些平方数的算术根，按升序输出。不需要多余的空格。如果有多种最优分拆，输出使前面的数尽量小的方案。

Sample Input

```
23
29
0
```

Sample Output

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

5．邮局选址问题

有一条公路经过V个村庄，每一个村庄都处在整数的坐标点上（这里假设公路拉直为X轴）。规划在这条公路上建立P个邮局，当然为了方便，这些邮局应建在某P个村庄上，但是要求让不同村庄的人到邮局要走的总路程最小。

输入

测试数据输入的第一行是村庄个数V和邮局个数P，第二行是V个村庄的坐标（坐标＞＝０）。1≤V，P≤1000，0≤村庄坐标x≤10000。

输出

输出分两行，第一行输出村庄到邮局距离总和的最小值，第二行按递增顺序输出 P 个邮局所在村庄的坐标。

Sample Input
10 5
1 2 6 8 10 13 18 23 33 40
Sample Output
15
2 5 7 9 10

6．投资问题

现有资金（设总数量为m）投入n个项目，并且给出投资收益表a[i][j]（均为整数），a[i][j]表示量为j（0≤j≤m）的资金投入项目i（0≤i<n）能获得的收益，请设计算法求出最大投资收益（注：资金不一定要用完）。

输入

数据首行两个整数值分别为m、n，紧接着是一个n行m+1列的矩阵，对应收益表a[][]，其中m<=100为整数，n<=20。

例如
8. 3

	0	1	2	3	4	5	6	7	8
0	0	5	15	40	80	90	95	98	100
1	0	5	15	40	60	70	73	74	75
2	0	4	26	40	45	50	51	52	53

（注：上面白色背景部分数据为a[][ ]的值，绿色背景部分只是助于理解）

输出

例如上面测试数据的最大投资收益值输出结果为
140

7．滑雪
http://poj.org/problem?id=1088

8．Zipper
http://poj.org/problem?id=2192

9．Multiplication Puzzle
http://poj.org/problem?id=1651

10．Palindrome
http://poj.org/problem?id=1159

二 回溯练习题

1.输出1~n的全排列。（时间：5分钟）

2.输出1~n中取m个的全排列。（时间：5分钟）

3.输出1~n的所有子集。（时间：6分钟）

4.输出1~n的包含k个元素的所有子集。（时间：8分钟）

5.输出1~n的和为S的子集（时间：8分钟）

6.输出1~n的含有k个元素且和为S的所有子集。（时间：10分钟）

7.将1~n分成k个互不相交、且并集为全集的非空子集，输出所有划分情况。（时间：30分钟）

例如n=4,k=3,即将U={1,2,3,4}划分成3个互不相交且并集为U的所有子集，划分结果有：{1,2}, {3}, {4}

{1,3}, {2}, {4}

{1}, {2,3}, {4}

{1,4}, {2}, {3}

{1}, {2,4}, {3}

{1}, {2}, {3,4}

（1,2为排列型，3,4,5,6,7为组合型）

参考解答：

一、动态规划练习题(vc6.0)

1. 解：设第n个月兔子总数为f(n)，则有

月 份　总数

n　　f(n)

n+1　f(n+1)

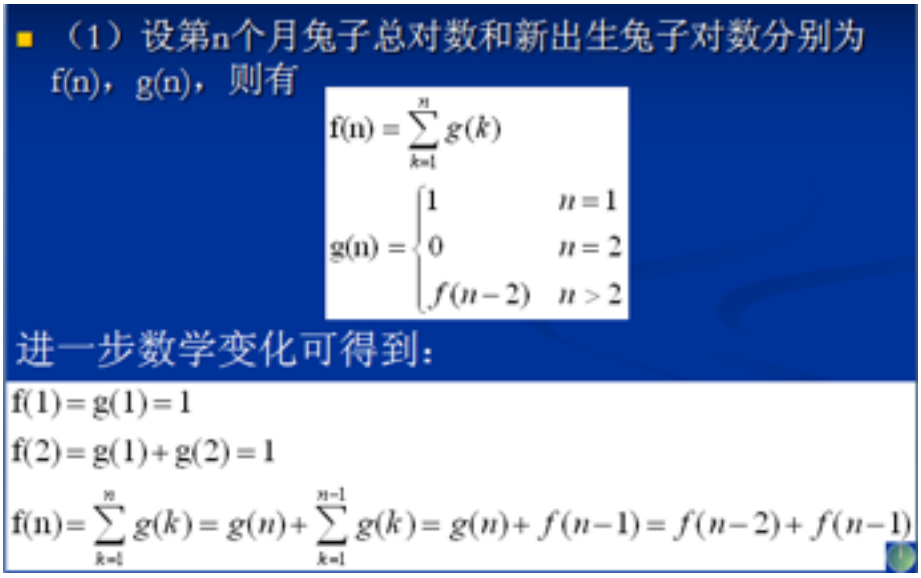
n+2　f(n+1)+2f(n) //老兔子+新生兔子

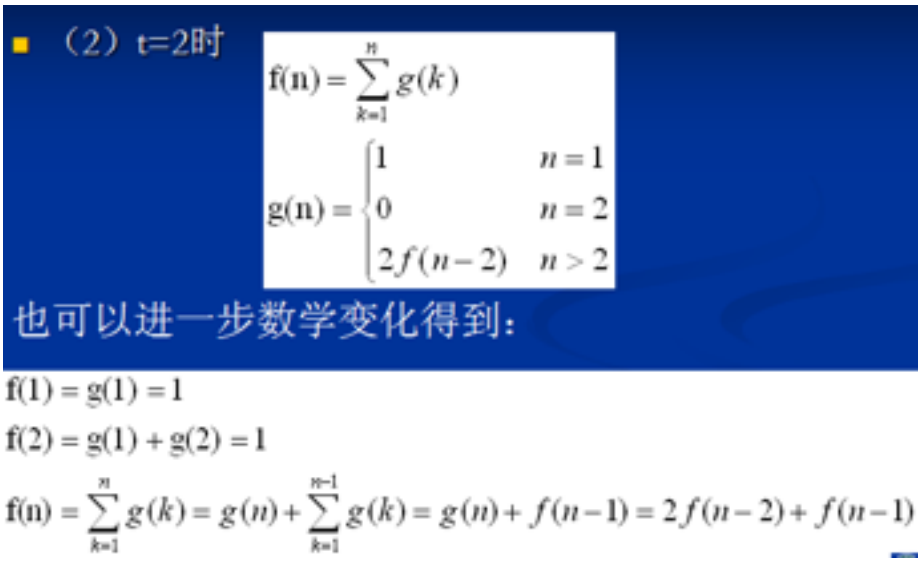
//老兔子：即前一个月份兔子总数f(n+1)

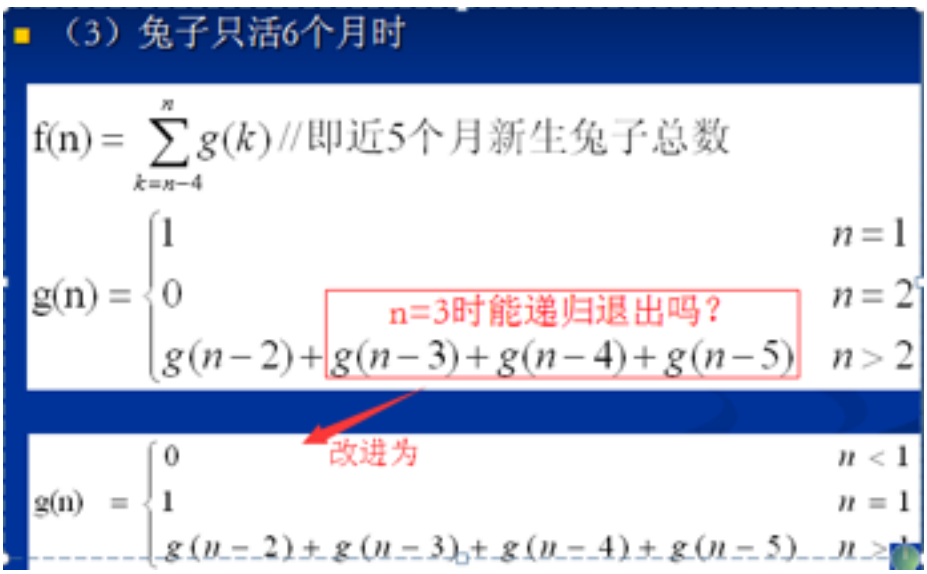
//新兔子：根据兔子繁殖规律，第n个月的所有兔子在第n+2个月均会繁殖，所以新生兔子数为2f(n)

即有f(n+2)=f(n+1)+2f(n)，且f(1)=f(2)=1;

当然这里的思考显得有点抽象，我们也可以用2个递推关系来分析。







计算f(n)的动态规划程序//也可以写两个动态规划函数分别计算f(n)和g(n)

```
#include<iostream>
#define MAX 100
using namespace std;

int F[MAX],n;

int f(int i)
{
    if(F[i]) return F[i];
    if(i==1 || i==2) return F[i]=1;
    else return F[i]=2*f(i-2)+f(i-1);
}

void main()
{
    int i=1;
    cout<<endl<<"Enter n:";
    cin>>n;
    while(i<=n)
    {
        cout<<endl<<i<<":"<<f(i);//i较大时会溢出，此程序没考虑溢出处理
        i++;
    }
}

//(3)参考程序
#include <iostream>
#define MAX 1000
using namespace std;

int F[MAX],G[MAX];//对应f(n),g(n)的存储
int f(int );
int g(int );

int f(int n)//求f(n)
{
    int k;
    if(n<1)  return 0;//注意此语句要放在最前面，因为n<0时不能访问F[n]
    if(F[n]) return F[n];//如果F[n]已经计算过，直接返回结果
    for(k=n-4;k<=n;k++) //近5个月
        F[n] += g(k);//对应f(n)=∑g(k) ， 1<=k<=n
    return F[n];
}

int g(int n)//根据g(n)等价的递推关系式而写
```

```
{
    if(n<1)   return 0;//异常，与前一个同理
    if(G[n])  return G[n];//如果G[n]已经计算过，直接返回结果
    if(n==1)  return G[n]=1;//保存结果并返回
    else      return G[n]=g(n-5)+g(n-4)+g(n-3)+g(n-2);//
}

int main()
{
    int n,i;
    cout<<"input n = ";
    cin>>n;

    for (i = 1; i <= n; i++)
    {
        cout<<"f("<<i<<")="<<f(i)<<"   g("<<i<<")="<<g(i)<<endl;
    }
    return 0;
}
```

2.解：第k天的植物总数用M(k)表示,f(k)表示第k天种子数量，则分析有

$$f(k)=\begin{cases} 1 & k==0 \\ 0 & k<x \\ \sum f(t) & k>=x,\text{其中}t=k-y,k-y+1,...,k-x \end{cases}$$

$$M(k)=\begin{cases} 0 & k==0 \\ 1 & k<=x \\ \sum f(t) & k>=x,\text{其中}t=k-z+1,...,k-1 \end{cases}$$

动态规划程序如下：

```
#include<iostream>
#define MAX 2501
using namespace std;

int x,y,z,n;
int F[MAX],M[MAX];
int f(int);
int m(int);

void Init( )
{
    int i;
    for(i=0;i<MAX;i++) F[i]=M[i]=-1;
```

```
}

int f(int k)//f(k)=∑f(t)   t=k-y,k-y+1,...,k-x
{
    int t;
    if(k<0)   return 0;
    if(F[k]!=-1)       return F[k];
     else if(k==0)      return F[k]=1;
    else if(k<x)       return F[k]=0;
    else
    {
        F[k]=0;
        for(t=k-y;t<=k-x;t++)
            F[k]=(F[k]+f(t))%100000000;
        return F[k];
    }
}

int m(int k)//M(k)=∑f(t)   t=k-z+1,...,k-1
{
    int t;
    if(M[k]!=-1)  return M[k];
    else if(k==0) return M[k]=0;
    else if(k<=x) return M[k]=1;
    else
    {
        M[k]=0;
        for(t=k-z+1;t<k;t++)
            M[k]=(M[k]+f(t))%100000000;
        return M[k];
    }
}

int main( )
{
    cin>>x>>y>>z>>n;
    while(x)
    {
        Init();
        cout<<m(n)<<endl;
        cin>>x>>y>>z>>n;
    }
    return 1;
}
```

3.解：f(s,t)表示用a1,a2,...,at面值硬币组成金额s，所用最少枚数。则有

s/a1　　t==1且s%a1==0

f(s,t)={ min{f(s-i*at,t-1)+i}　t>1 且0<=i<=s/at

【注：此问题中一般默认a1=1,保证一定有解;否则递推关系式需要修正如下】

s/a1　　t==1且s%a1==0

f(s,t)={ -1　　t==1且s%a1!=0

min{f(s-i*at,t-1)+i}　t>1 且0<=i<=s/at，f(s-i*at)!=-1

【注：用-1表示无解时f(s,t)返回的值】

前一种递推式的动态规划程序

```
#include<iostream>
#define MAX 1001
using namespace std;

int F[MAX][MAX],n,k,a[MAX];

void init()
{
    int j;
    memset(F,0,MAX*MAX*sizeof(int));
    cin>>k>>n;
    for(j=1;j<=k;j++)
        cin>>a[j];
}

int f(int s,int t)
{
    int min=s,i,tmp;
    if(F[s][t]) return F[s][t];
    if(t==1 && s%a[t]==0)      return F[s][t]=s/a[t];
    else
    {
        for(i=0;i<=s/a[t];i++)
            if((tmp=f(s-i*a[t],t-1)+i) < min) min=tmp;
        return F[s][t]=min;
    }
}

int main()
{
    init();
    while(k)
    {
        cout<<f(n,k)<<endl;
```

```
            init();
        }
        return 1;
}
```

4.解：用f(s,t)表示将s分解为大于等于t的数字的平方和，最少数字个数，则有

$$f(s,t)=\begin{cases} s+1 & t*t>s \quad //\text{无解情况} \\ 1 & t*t==s \\ \min\{f(s,t+1),f(s-t*t,t)+1\} & t*t<s \end{cases}$$

动态规划程序

```
#include<iostream>
#define MAX 100001
using namespace std;

int F[MAX][320],n;//320的平方已经超出100000

void init()
{
    int j;
    memset(F,0,MAX*320*sizeof(int));
    cin>>n;
}

int min(int a,int b)
{
    return a<b?a:b;
}

int f(int s,int t)
{
    if(F[s][t])    return F[s][t];
    if(t*t>s)      return F[s][t]=s+1;
    else if(t*t==s)return F[s][t]=1;
    else           return F[s][t]=min(f(s,t+1),f(s-t*t,t)+1);
}

void calcT(int s,int t)
{
    if(t*t==s){ cout<<t<<" "; return ;}
    if(F[s-t*t][t]+1==F[s][t])
    {
        cout<<t<<" ";
        calcT(s-t*t,t);
```

```
    }
    else calcT(s,t+1);

}

int main()
{
    init();
    while(n)
    {
        cout<<f(n,1)<<endl;
        calcT(n,1);
        cout<<endl;
        init();
    }
    return 1;
}
```

5.解：用f(V,P)表示V个村庄(村庄依次编号1,2,...,V)P个邮局时，村庄到邮局距离总和的最小值，

用len(i,j)表示在i~j村庄范围设置一个邮局时各村庄到邮局的距离和最小值，可以分析出

邮局安排在编号为(i+j)/2的村庄时，其距离和达到最小值。

可分析出递推关系式如下

$$f(V,P)=\begin{cases} 0 & V<=P \quad \text{//邮局个数多余村庄个数} \\ len(1,V) & P=1 \quad \text{//仅一个邮局时} \\ \min\{f(V-k,P-1)+len(V-k+1,V)\} & V>P>1 \quad \text{//最右边邮局管辖村庄范围V-k+1~V} \end{cases}$$

【其中k满足k>=1且V-k>=P-1，即1<=k<=V-P+1】

动态规划程序

```
#include<iostream>
#define MAX 1001
using namespace std;//

int a[MAX],len[MAX][MAX],F[MAX][MAX],V,P;

void init()
{
    int i,j,mid;
    memset(F,-1,MAX*MAX*sizeof(int));
    cin>>V>>P;
    for(i=1;i<=V;i++) cin>>a[i];
    for(i=1;i<=V;i++)
    {
```

```
            len[i][i]=0;
            for(j=i+1;j<=V;j++)
            {
                mid=(i+j)/2;
                len[i][j]=len[i][j-1]+a[j]-a[mid];
            }
        }
}

int f(int V,int P)
{
        int min,k,tmp;
        if(F[V][P]!=-1) return F[V][P];
        if(V<=P)        return F[V][P]=0;
        else if(P==1)   return F[V][P]=len[1][V];
        else
        {
            min=len[1][V];
            for(k=1;k<=V-P+1;k++)
                if((tmp=f(V-k,P-1)+len[V-k+1][V]) < min) min=tmp;
            return F[V][P]=min;
        }
}

void calcX(int V,int P)
{
        int k;
        if(P==1)
        {
            cout<<(V+1)/2<<" ";
            return ;
        }
        for(k=1;k<=V-P+1;k++)
          if(F[V][P]==F[V-k][P-1]+len[V-k+1][V]) break;
        calcX(V-k,P-1);
        cout<<V-k/2<<" ";
}

int main()
{
        init();
        cout<<f(V,P)<<endl;
        calcX(V,P);
        return 1;
}
```

6.解：分析递推关系式，用f(i,j)表示资金量为j项目范围为1~i，能够获得的最大投资收益

则有：

$$f(i,j)=\begin{cases}0 & i<0 \ || \ j==0\\ \max\{a[i][k]\} & i==0,\text{其中}0<=k<=j\\ \max\{a[i][k]+f(i-1,j-k)\} & i>0,\text{其中}0<=k<=j\end{cases}$$

动态规划程序

```
#include<iostream>
using namespace std;
#define MAX 101

int a[MAX][MAX],F[MAX][MAX],m,n;

void init()
{
    int i,j;
    memset(F,-1,MAX*MAX*sizeof(int));
    cin>>m>>n;
    for(i=0;i<n;i++)
      for(j=0;j<=m;j++)
          cin>>a[i][j];
}

int f(int i,int j)
{
    int mmax,k,tmp;
    if(F[i][j]!=-1) return F[i][j];
    if(i<0)      return 0;
    else if(j==0) return F[i][j]=0;
    else if(i==0)
    {
        mmax=a[i][0];
        for(k=1;k<=j;k++)
            if(a[i][k]>mmax) mmax=a[i][k];
        return F[i][j]=mmax;
    }
    else
    {
        mmax=a[i][0]+f(i-1,j);
        for(k=1;k<=j;k++)
        {
            tmp=a[i][k]+f(i-1,j-k);
```

```
                if(tmp>mmax) mmax=tmp;
            }
            return F[i][j]=mmax;
        }
}

int main()
{
    init();
    cout<<f(n,m);
    return 1;
}
```

7.

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

int c,r,MaxInt=(int)pow(2,sizeof(int)-1)-1;//行,列
int h[102][102];//二维高度数组
int cc[102][102];//二维存储数组


int Max(int a,int b,int c,int d){
    int max=1;
    if(a>max) max=a;
    if(b>max) max=b;
    if(c>max) max=c;
    if(d>max) max=d;
    return max;
}

int Min(int a,int b,int c,int d){
    int max=(int)pow(2,sizeof(int)-1)-1;
    if(a<max) max=a;
    if(b<max) max=b;
    if(c<max) max=c;
    if(d<max) max=d;
    return max;
}

int f(int i,int j){
```

```
    if(cc[i][j]!=-1) return cc[i][j];
    else if(h[i][j]<Min(h[i-1][j],h[i+1][j],h[i][j-1],h[i][j+1])) return cc[i][j]=1;
    else{
        int a=0,b=0,c=0,d=0;
        if(h[i][j]>h[i-1][j]) a=f(i-1,j)+1;
        if(h[i][j]>h[i][j-1]) b=f(i,j-1)+1;
        if(h[i][j]>h[i+1][j]) c=f(i+1,j)+1;
        if(h[i][j]>h[i][j+1]) d=f(i,j+1)+1;

        return cc[i][j]=Max(a,b,c,d);
    }
}

int high(int i,int j)//判断(i,j)是否比周围的点要高，即局部高点
{
    if(i>1 && h[i][j] < h[i-1][j]) return 0;
    if(i<c && h[i][j] < h[i+1][j]) return 0;
    if(j>1 && h[i][j] < h[i][j-1]) return 0;
    if(j<r && h[i][j] < h[i][j+1]) return 0;
    return 1;
}

int main(){
    int i,j;
    cin>>c>>r;
    for(i=1;i<c+1;i++)
        for(j=1;j<r+1;j++)
        {
            cc[i][j]=-1;
            cin>>h[i][j];
        }
    for(i=0;i<r+2;i++) {h[0][i]=MaxInt; h[c+1][i]=MaxInt;}
    for(i=0;i<c+2;i++) {h[i][0]=MaxInt; h[i][r+1]=MaxInt;}

    int max=1;
    for(i=1;i<c+1;i++)
        for(j=1;j<r+1;j++)
          if(high(i,j) && max<f(i,j)) max=cc[i][j];

    cout<<max;

    return 0;
}
```

8.

9.
10.

二、回溯练习题(vc6.0)
1.

```
#include<iostream>
#define MAX 100
using namespace std;

int n,X[MAX];

int xianjie(int k,int i)
{
  int j;
  for(j=1;j<k;j++)
     if(X[j]==i) return 0;
  return 1;
}

void f(int k)
{
  int i;
  if(k==n+1)
  {  cout<<endl;
    for(i=1;i<=n;i++) cout<<X[i]<<" ";
   }
  else
    for(i=1;i<=n;i++)
      if(xianjie(k,i))
       {
         X[k]=i;
         f(k+1);
       }
}

void main()
{
  cout<<endl<<"Enter n(<"<<MAX<<"):";
  cin>>n;
  f(1);
}
```

2.

```
#include<iostream>
#define MAX 100
```

```
using namespace std;

int n,m,X[MAX];

int xianjie(int k,int i)
{
  int j;
  for(j=1;j<k;j++)
     if(X[j]==i) return 0;
  return 1;
}

void f(int k)
{
  int i;
  if(k==m+1)
  {  cout<<endl;
    for(i=1;i<=m;i++) cout<<X[i]<<" ";
   }
  else
    for(i=1;i<=n;i++)
      if(xianjie(k,i))
       {
         X[k]=i;
         f(k+1);
       }
}

void main()
{
  cout<<endl<<"Enter n,m(<"<<MAX<<",m<=n):";
  cin>>n>>m;
  f(1);
}
```

3.

```
#include<iostream>
#define MAX 100
using namespace std;

int n,X[MAX];

int xianjie(int k,int i)
{
     return 1;
}
```

```
void f(int k)
{
    int i;
    if(k==n+1)
    {
        cout<<endl<<"{";
        for(i=1;i<=n;i++)
            if(X[i]==1) cout<<i<<" ";
        cout<<"}";
     }
    else
        for(i=1;i<=2;i++)
            if(xianjie(k,i))
            {
                X[k]=i;
                f(k+1);
            }
}

void main()
{
    cout<<endl<<"Enter n(<"<<MAX<<"):";
    cin>>n;
    f(1);
}
```

4.

```
#include<iostream>
#define MAX 100
using namespace std;

int n,k,X[MAX];
int count=0;

int xianjie(int t,int i)
{
    if(i==1 && count+1>k)   return 0;
    if(i==2 && count+n-t<k) return 0;
    return 1;
}

void f(int t)
{
    int i;
    if(t==n+1)
```

```
    {
        cout<<endl<<"{";
        for(i=1;i<=n;i++)
            if(X[i]==1) cout<<i<<" ";
        cout<<"}";
    }
    else
        for(i=1;i<=2;i++)
            if(xianjie(t,i))
            {
                X[t]=i;
                if(i==1) count++;
                f(t+1);
                if(i==1) count--;
            }
}

void main()
{
    cout<<endl<<"Enter n,k(<"<<MAX<<"k<=n):";
    cin>>n>>k;
    f(1);
}
```

5.

```
#include<iostream>
#define MAX 100
using namespace std;

int n,S,X[MAX];
int curS,leftS;

int xianjie(int t,int i)
{
    if(i==1 && curS+t>S)       return 0;
    if(i==2 && curS+leftS-t<S) return 0;
    return 1;
}

void f(int t)
{
    int i;
    if(t==n+1 && curS==S)
    {
        cout<<endl<<"{";
        for(i=1;i<=n;i++)
```

```
                if(X[i]==1) cout<<i<<" ";
            cout<<"}";
        }
    else
        for(i=1;i<=2;i++)
            if(xianjie(t,i))
            {
                X[t]=i;
                if(i==1) curS += t;
                leftS -= t;
                f(t+1);
                if(i==1) curS -= t;
                leftS += t;
            }
}

void main()
{
    cout<<endl<<"Enter n,S(<"<<MAX<<"):";
    cin>>n>>S;
    curS=0;
    leftS=n*(n+1)/2;
    f(1);
}
```

6.

```
#include<iostream>
#define MAX 100
using namespace std;

int n,k,S,X[MAX];
int curS,leftS,count=0;

int xianjie(int t,int i)
{
    if(i==1 && curS+t>S)       return 0;
    if(i==2 && curS+leftS-t<S) return 0;
    if(i==1 && count+1>k)      return 0;
    if(i==2 && count+n-t<k)    return 0;
    return 1;
}

void f(int t)
{
    int i;
    if(t==n+1 && curS==S && count==k)
```

```
    {
        cout<<endl<<"{";
        for(i=1;i<=n;i++)
            if(X[i]==1) cout<<i<<" ";
        cout<<"}";
    }
    else
        for(i=1;i<=2;i++)
            if(xianjie(t,i))
            {
                X[t]=i;
                if(i==1) { curS += t; count++;}
                leftS -= t;
                f(t+1);
                if(i==1) { curS -= t; count--;}
                leftS += t;
            }
}

void main()
{
    cout<<endl<<"Enter n,k,S(<"<<MAX<<"k<=n):";
    cin>>n>>k>>S;
    curS=0;
    leftS=n*(n+1)/2;
    f(1);
}
```

7.

```
#include<iostream>
#define MAX 100
using namespace std;

int n,k,X[MAX],count[MAX];

int xianjie(int t,int i)
{
    int j;
    if(i>1 && !count[i-1])   return 0;
    if(t==n)
    {
        for(j=1;j<=k;j++)
            if(j!=i && !count[j]) return 0;
    }
    return 1;
}
```

```
void f(int t)
{
	int i,j;
	if(t==n+1)
	{
		cout<<endl;
		for(j=1;j<=k;j++)
		{
			cout<<"{";
			for(i=1;i<=n;i++)
				if(X[i]==j) cout<<i<<" ";
			cout<<"}";
		}
	 }
	else
		for(i=1;i<=k;i++)
			if(xianjie(t,i))
			{
				X[t]=i;
				count[i]++;
				f(t+1);
				count[i]--;
			}
}

void main()
{
	cout<<endl<<"Enter n,k(<"<<MAX<<"k<=n):";
	cin>>n>>k;
	f(1);
}
```