**Problem A**

**Problem Description**

Given a sequence a[1],a[2],a[3]......a[n], your job is to calculate the max sum of a sub-sequence. For example, given (6,-1,5,4,-7), the max sum in this sequence is 6 + (-1) + 5 + 4 = 14.

**Input**

The first line of the input contains an integer T(1<=T<=20) which means the number of test cases. Then T lines follow, each line starts with a number N(1<=N<=100000), then N integers followed(all the integers are between -1000 and 1000).

**Output**

For each test case, you should output two lines. The first line is "Case #:", # means the number of the test case. The second line contains three integers, the Max Sum in the sequence, the start position of the sub-sequence, the end position of the sub-sequence. If there are more than one result, output the first one. Output a blank line between two cases.

**Sample Input**

2

5 6 -1 5 4 -7

7 0 6 -1 1 -6 7 -5

**Sample Output**

Case 1:

14 1 4

Case 2:

7 1 6

**代码：**

#include<stdio.h>

#include<iostream>

using namespace std;

int max(int a,int b){

return a>b?a:b;

}

int main(){

int a,b,t=0;

int c[100000];

int f[100000];

int p[100000];

cin>>a;

while(a--){

t++;

cin>>b;

for(int i=0;i<b;i++){

cin>>c[i];

p[i]=i;

}

f[0]=c[0];

for(int i=1;i<b;i++){

f[i]=max(c[i],f[i-1]+c[i]);

if(f[i]==c[i])p[i]=i;

if(f[i]==c[i]+f[i-1])p[i]=p[i-1];

}

int Max=f[0];

int s=p[0];

int e=0;

for(int i=1;i<b;i++){

if(Max<f[i]){

Max=f[i];

s=p[i];

e=i;

}

}

cout<<"Case "<<t<<":"<<endl<<Max<<" "<<s+1<<" "<<e+1<<endl;

if(a)cout<<endl;

}

return 0;

}

**Problem B**

**Problem Description**

平面上有n条直线，且无三线共点，问这些直线能有多少种不同交点数。  
比如,如果n=2,则可能的交点数量为0(平行)或者1(不平行)。

**Input**

输入数据包含多个测试实例,每个测试实例占一行,每行包含一个正整数n（n<=20）,n表示直线的数量.

**Output**

每个测试实例对应一行输出，从小到大列出所有相交方案，其中每个数为可能的交点数,每行的整数之间用一个空格隔开。

**Sample Input**

2

3

**Sample Output**

0 1

0 2 3

**代码：**

#include<stdio.h>

int main()

{

int i,n,st[21][200],j,k;

for(i=1;i<=20;i++)

for(j=0;j<=200;j++)

st[i][j]=0;

st[0][0]=1;

st[1][0]=1;

st[2][0]=1;

st[2][1]=1;

for(i=3;i<=20;i++)

for(j=0;j<i;j++)

for(k=0;k<=(j-1)\*j/2;k++)

if(st[j][k]==1)

st[i][k+j\*(i-j)]=1;

while(scanf("%d",&n)!=EOF)

{

printf("0");

for(i=1;i<=(n-1)\*n/2;i++)

if(st[n][i]==1)

printf(" %d",i);

printf("\n");

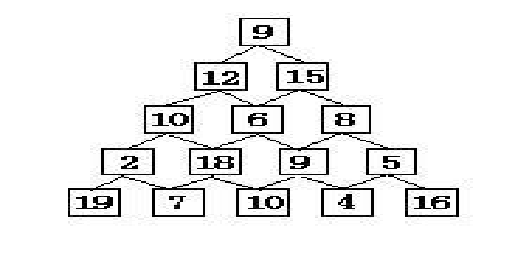
}

return 0;

}

**Problem C**

**Problem Description**

在讲述DP算法的时候，一个经典的例子就是数塔问题，它是这样描述的：  
  
有如下所示的数塔，要求从顶层走到底层，若每一步只能走到相邻的结点，则经过的结点的数字之和最大是多少？  
  
已经告诉你了，这是个DP的题目，你能AC吗?

**Input**

输入数据首先包括一个整数C,表示测试实例的个数，每个测试实例的第一行是一个整数N(1 <= N <= 100)，表示数塔的高度，接下来用N行数字表示数塔，其中第i行有个i个整数，且所有的整数均在区间[0,99]内。

**Output**

对于每个测试实例，输出可能得到的最大和，每个实例的输出占一行。

**Sample Input**

1

5

7

3 8

8 1 0

2 7 4 4

4 5 2 6 5

**Sample Output**

30

**代码：**

#include<stdio.h>

int max(int a,int b){

return a>b?a:b;

}

int main(){

int a,b;

int i,j,k;

scanf("%d",&a);

while(a!=0){

scanf("%d",&b);

int arr[b][b];

for(j=0;j<b;j++){

for(k=0;k<j+1;k++){

scanf("%d",&arr[j][k]);

}

}

for(i=b-2;i>=0;i--){

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

arr[i][j]+=max(arr[i+1][j],arr[i+1][j+1]);

}

}

printf("%d\n",arr[0][0]);

a--;

}

return 0;

}

**Problem D**

**Problem Description**

FatMouse believes that the fatter a mouse is, the faster it runs. To disprove this, you want to take the data on a collection of mice and put as large a subset of this data as possible into a sequence so that the weights are increasing, but the speeds are decreasing.

**Input**

Input contains data for a bunch of mice, one mouse per line, terminated by end of file.  
  
The data for a particular mouse will consist of a pair of integers: the first representing its size in grams and the second representing its speed in centimeters per second. Both integers are between 1 and 10000. The data in each test case will contain information for at most 1000 mice.  
  
Two mice may have the same weight, the same speed, or even the same weight and speed.

**Output**

Your program should output a sequence of lines of data; the first line should contain a number n; the remaining n lines should each contain a single positive integer (each one representing a mouse). If these n integers are m[1], m[2],..., m[n] then it must be the case that   
  
W[m[1]] < W[m[2]] < ... < W[m[n]]  
  
and   
  
S[m[1]] > S[m[2]] > ... > S[m[n]]  
  
In order for the answer to be correct, n should be as large as possible.  
All inequalities are strict: weights must be strictly increasing, and speeds must be strictly decreasing. There may be many correct outputs for a given input, your program only needs to find one.

**Sample Input**

6008 1300

6000 2100

500 2000

1000 4000

1100 3000

6000 2000

8000 1400

6000 1200

2000 1900

**Sample Output**

4

4

5

9

7

**代码：**

#include<stdio.h>

#include<algorithm>

using namespace std;

typedef struct{

int w;

int s;

int e;

}Mouse;

bool cmp(Mouse a,Mouse b){

if(a.w==b.w)return a.s<b.s;

else{

return a.w>b.w;

}

}

int max(int a,int b){

return a>b?a:b;

}

int main(){

int count=-1;

int i=0;

int j=0;

Mouse mouse[1000];

while(scanf("%d %d",&mouse[i].w,&mouse[i].s)!=EOF){

mouse[i].e=i+1;

count++;

i++;

}

sort(mouse,mouse+count+1,cmp);

int f[1000]={0};

f[0]=1;

int big=0;

for(i=1;i<count;i++){

big=0;

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

if(mouse[i].s>mouse[j].s&&mouse[i].w!=mouse[j].w){

big=max(f[j],big);

}

}

f[i]=big+1;

}

big=0;

int bignum=0;

int bigcount=0;

for(i=0;i<count;i++){

if(f[i]>big){

big=f[i];

bignum=mouse[i].e;

bigcount=i;

}

}

printf("%d\n",big);

printf("%d\n",bignum);

while(big>0){

big--;

for(i=bigcount;i>=0;i--){

if(f[i]==big){

printf("%d\n",mouse[i].e);

bigcount=i;

break;

}

}

}

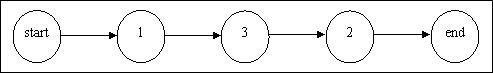
return 0;

}

**Problem E**

**Problem Description**

Nowadays, a kind of chess game called “Super Jumping! Jumping! Jumping!” is very popular in HDU. Maybe you are a good boy, and know little about this game, so I introduce it to you now.



The game can be played by two or more than two players. It consists of a chessboard（棋盘）and some chessmen（棋子）, and all chessmen are marked by a positive integer or “start” or “end”. The player starts from start-point and must jumps into end-point finally. In the course of jumping, the player will visit the chessmen in the path, but everyone must jumps from one chessman to another absolutely bigger (you can assume start-point is a minimum and end-point is a maximum.). And all players cannot go backwards. One jumping can go from a chessman to next, also can go across many chessmen, and even you can straightly get to end-point from start-point. Of course you get zero point in this situation. A player is a winner if and only if he can get a bigger score according to his jumping solution. Note that your score comes from the sum of value on the chessmen in you jumping path.  
Your task is to output the maximum value according to the given chessmen list.

**Input**

Input contains multiple test cases. Each test case is described in a line as follow:  
N value\_1 value\_2 …value\_N   
It is guarantied that N is not more than 1000 and all value\_i are in the range of 32-int.  
A test case starting with 0 terminates the input and this test case is not to be processed.

**Output**

For each case, print the maximum according to rules, and one line one case.

**Sample Input**

3 1 3 2

4 1 2 3 4

4 3 3 2 1

0

**Sample Output**

4

10

3

**代码：**

#include<stdio.h>

int max(int a,int b){

return a>b?a:b;

}

int main(){

int i,j;

int b;

int temp;

scanf("%d",&b);

\_\_int32 a[1001];

while(b!=0){

for(i=0;i<b;i++)scanf("%I32d",&a[i]);

\_\_int32 f[1001];

f[0]=a[0];

for(i=1;i<b;i++){

f[i]=a[i];

temp=0;

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

if(a[j]<a[i]){

if(f[j]+a[i]>f[i])f[i]=f[j]+a[i];

}

}

}

int Max=f[0];

for(i=1;i<b;i++){

Max=max(Max,f[i]);

}

printf("%d\n",Max);

scanf("%d",&b);

}

return 0;

}

**Problem F**

**Problem Description**

A subsequence of a given sequence is the given sequence with some elements (possible none) left out. Given a sequence X = <x1, x2, ..., xm> another sequence Z = <z1, z2, ..., zk> is a subsequence of X if there exists a strictly increasing sequence <i1, i2, ..., ik> of indices of X such that for all j = 1,2,...,k, xij = zj. For example, Z = <a, b, f, c> is a subsequence of X = <a, b, c, f, b, c> with index sequence <1, 2, 4, 6>. Given two sequences X and Y the problem is to find the length of the maximum-length common subsequence of X and Y.   
The program input is from a text file. Each data set in the file contains two strings representing the given sequences. The sequences are separated by any number of white spaces. The input data are correct. For each set of data the program prints on the standard output the length of the maximum-length common subsequence from the beginning of a separate line.

**Sample Input**

abcfbc abfcab

programming contest

abcd mnp

**Sample Output**

4

2

0

**代码：**

#include<stdio.h>

#include<string.h>

int max(int a,int b){

return a>b?a:b;

}

int main(){

char s1[700];

char s2[700];

int f[700][700]={

0

};

while(scanf("%s%s",s1,s2)!=EOF){

int m=strlen(s1);

int n=strlen(s2);

for(int i=0;i<m;i++){

for(int j=0;j<n;j++){

if(s1[i]==s2[j]){

f[i+1][j+1]=f[i][j]+1;

}else{

f[i+1][j+1]=max(f[i][j+1],f[i+1][j]);

}

}

}

int max=0;

for(int i=0;i<m;i++){

for(int j=0;j<n;j++){

if(f[i+1][j+1]>max){

max=f[i+1][j+1];

}

}

}

printf("%d\n",max);

}

return 0;

}

**Problem G**

**Problem Description**

Yifenfei very like play a number game in the n\*n Matrix. A positive integer number is put in each area of the Matrix.  
Every time yifenfei should to do is that choose a detour which frome the top left point to the bottom right point and than back to the top left point with the maximal values of sum integers that area of Matrix yifenfei choose. But from the top to the bottom can only choose right and down, from the bottom to the top can only choose left and up. And yifenfei can not pass the same area of the Matrix except the start and end.

**Input**

The input contains multiple test cases.  
Each case first line given the integer n (2<n<30)   
Than n lines,each line include n positive integers.(<100)

**Output**

For each test case output the maximal values yifenfei can get.

**Sample Input**

2

10 3

5 10

3

10 3 3

2 5 3

6 7 10

5

1 2 3 4 5

2 3 4 5 6

3 4 5 6 7

4 5 6 7 8

5 6 7 8 9

**Sample Output**

28

46

80

**代码：**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define s 30

int dp[s][s][s][s];

int vis[s][s][s][s];

int n;

int map[s][s];

int max(int a,int b,int c,int d)

{

if(a>=b&&a>=c&&a>=d)

return a;

if(b>=a&&b>=c&&b>=d)

return b;

if(c>=a&&c>=b&&c>=d)

return c;

if(d>=a&&d>=b&&d>=c)

return d;

}

int dfs(int t,int r,int p,int q)

{

int i,j,k,l;

if(t<0||t>=n||r<0||r>=n||p<0||p>=n||q<0||q>=n)

return 0;

if(vis[t][r][p][q]==1)

return dp[t][r][p][q];

vis[t][r][p][q]=1;

dp[t][r][p][q]=max(dfs(t-1,r,p-1,q),dfs(t-1,r,p,q-1),dfs(t,r-1,p-1,q),dfs(t,r-1,p,q-1))+map[t][r]+map[p][q];

vis[t][r][p][q]=1;

return dp[t][r][p][q];

}

int main()

{

int i,j,k,l;

while(scanf("%d",&n)!=EOF)

{

for(i=0;i<n;i++)

for(j=0;j<n;j++)

scanf("%d",&map[i][j]);

memset(dp,0,sizeof(dp));

memset(vis,0,sizeof(vis));

dp[0][0][0][0]=0;

vis[0][0][0][0]=1;

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

dp[i][j][i][j]=0;

vis[i][j][i][j]=1;

}

}

printf("%d\n",dfs(n-1,n-2,n-2,n-1)+map[0][0]+map[n-1][n-1]);

}

return 0;

}