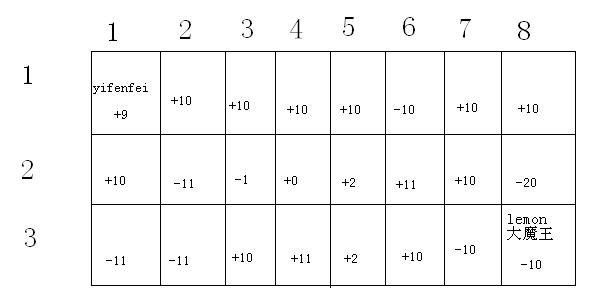
**命运**

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 16179    Accepted Submission(s): 5652**

Problem Description

穿过幽谷意味着离大魔王lemon已经无限接近了！  
可谁能想到，yifenfei在斩杀了一些虾兵蟹将后，却再次面临命运大迷宫的考验，这是魔王lemon设下的又一个机关。要知道，不论何人，若在迷宫中被困1小时以上，则必死无疑！  
可怜的yifenfei为了去救MM，义无返顾地跳进了迷宫。让我们一起帮帮执着的他吧！  
命运大迷宫可以看成是一个两维的方格阵列，如下图所示：  
  
yifenfei一开始在左上角，目的当然是到达右下角的大魔王所在地。迷宫的每一个格子都受到幸运女神眷恋或者痛苦魔王的诅咒，所以每个格子都对应一个值，走到那里便自动得到了对应的值。  
现在规定yifenfei只能向右或者向下走，向下一次只能走一格。但是如果向右走，则每次可以走一格或者走到该行的列数是当前所在列数倍数的格子，即：如果当前格子是（x,y），下一步可以是（x+1,y），(x,y+1)或者(x,y\*k) 其中k>1。   
为了能够最大把握的消灭魔王lemon，yifenfei希望能够在这个命运大迷宫中得到最大的幸运值。

Input

输入数据首先是一个整数C，表示测试数据的组数。  
每组测试数据的第一行是两个整数n,m，分别表示行数和列数(1<=n<=20,10<=m<=1000)；  
接着是n行数据，每行包含m个整数，表示n行m列的格子对应的幸运值K ( |k|<100 )。

Output

请对应每组测试数据输出一个整数，表示yifenfei可以得到的最大幸运值。

Sample Input

1

3 8

9 10 10 10 10 -10 10 10

10 -11 -1 0 2 11 10 -20

-11 -11 10 11 2 10 -10 -10

Sample Output

52

Author

yifenfei

#include <iostream>  
#include<memory.h>  
using namespace std;  
int max(int a,int b)  
{  
    if(a>b)  
    return a;  
    else  
    return b;  
}  
int  set[21][1001];  
int sum[21][1001];  
int main()  
{  
    int n;  
    cin>>n;  
    while(n--)  
    {  
        int a,b;  
        cin>>a>>b;  
        memset(sum,0,sizeof(sum));  
        int i,j;  
        for(i=1;i<=a;i++)  
        for(j=1;j<=b;j++)  
        cin>>set[i][j];  
        for(i=0;i<=a;i++)  
        sum[i][0]=-1000;  
        for(j=0;j<=b;j++)  
        sum[0][j]=-1000;  
        sum[0][1]=sum[1][0]=0;  
        for(i=1;i<=a;i++)  
        for(j=1;j<=b;j++)  
        {  
            int t,y,m=-1000;  
            for(t=2;t<=j;t++)  
              if(j%t==0)  
              { y=max(sum[i-1][j],sum[i][j/t]);  
                if(y>m)  
                 m=y;  
              }  
              y=max(sum[i-1][j],sum[i][j-1]);  
              if(y>m)  
                m=y;  
            sum[i][j]=m+set[i][j];  
             //cout<<sum[i][j]<<" ";  
             //cout<<endl;  
            //if(sum[i][j]>maxs)  
              //maxs=sum[i][j];  
        }

        cout <<sum[a][b]<< endl;  
    }

    return 0;  
}

**Monkey and Banana**

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 12566    Accepted Submission(s): 6567**

Problem Description

A group of researchers are designing an experiment to test the IQ of a monkey. They will hang a banana at the roof of a building, and at the mean time, provide the monkey with some blocks. If the monkey is clever enough, it shall be able to reach the banana by placing one block on the top another to build a tower and climb up to get its favorite food.  
  
The researchers have n types of blocks, and an unlimited supply of blocks of each type. Each type-i block was a rectangular solid with linear dimensions (xi, yi, zi). A block could be reoriented so that any two of its three dimensions determined the dimensions of the base and the other dimension was the height.   
  
They want to make sure that the tallest tower possible by stacking blocks can reach the roof. The problem is that, in building a tower, one block could only be placed on top of another block as long as the two base dimensions of the upper block were both strictly smaller than the corresponding base dimensions of the lower block because there has to be some space for the monkey to step on. This meant, for example, that blocks oriented to have equal-sized bases couldn't be stacked.   
  
Your job is to write a program that determines the height of the tallest tower the monkey can build with a given set of blocks.

Input

The input file will contain one or more test cases. The first line of each test case contains an integer n,  
representing the number of different blocks in the following data set. The maximum value for n is 30.  
Each of the next n lines contains three integers representing the values xi, yi and zi.  
Input is terminated by a value of zero (0) for n.

Output

For each test case, print one line containing the case number (they are numbered sequentially starting from 1) and the height of the tallest possible tower in the format "Case case: maximum height = height".

Sample Input

1

10 20 30

2

6 8 10

5 5 5

7

1 1 1

2 2 2

3 3 3

4 4 4

5 5 5

6 6 6

7 7 7

5

31 41 59

26 53 58

97 93 23

84 62 64

33 83 27

0

Sample Output

Case 1: maximum height = 40

Case 2: maximum height = 21

Case 3: maximum height = 28

Case 4: maximum height = 342

Source

[University of Ulm Local Contest 1996](http://acm.hdu.edu.cn/search.php?field=problem&key=University+of+Ulm+Local+Contest+1996&source=1&searchmode=source)

Recommend

JGShining

#include <iostream>  
#include<memory.h>  
#include <stdlib.h>  
#include<stdio.h>  
#include<algorithm>  
#define maxsim 200  
using namespace std;  
int max(int a,int b)  
{  
    if(a>b)  
    return a;  
    else  
    return b;  
}  
struct block  
{  
    int x,y,z;

}a[maxsim];

int compare(block a,block b)  
{  
 if(a.x!=b.x)  
 return a.x<b.x;  
    else  
        return a.y<=b.y;

}  
int main()  
{  
    int n,xx[3],num,d=0;  
    while(cin>>n&&n)  
    {  
        int i,dp[maxsim];  
        num=0;  
        for(i=0;i<n;i++)  
        {

            scanf("%d%d%d",&xx[0],&xx[1],&xx[2]);

             if(xx[2]==xx[0]&&xx[0]==xx[3])  
             {  
                a[num].x=xx[0];  
                a[num].y=xx[0];  
                a[num].z=xx[0];  
                num++;

             }  
             else  
             {  
                a[num].x=xx[0];  
                a[num].y=xx[1];  
                a[num].z=xx[2];  
                num++;  
                a[num].x=xx[1];  
                a[num].y=xx[2];  
                a[num].z=xx[0];  
                num++;

                a[num].x=xx[0];  
                a[num].y=xx[2];  
                a[num].z=xx[1];  
                num++;

                a[num].x=xx[1];  
                a[num].y=xx[0];  
                a[num].z=xx[2];

                num++;

                a[num].x=xx[2];  
                a[num].y=xx[0];  
                a[num].z=xx[1];  
                num++;

                a[num].x=xx[2];  
                a[num].y=xx[1];  
                a[num].z=xx[0];  
                num++;  
             }  
        }  
        memset(dp,0,sizeof(dp));  
        sort(a,a+num,compare);

        int m,j;  
        dp[0]=a[0].z;  
        m=0;  
        for(i=1;i<num;i++)  
          {  
              int oo=0;  
               for(j=i-1;j>=0;j--)  
               {  
                   if(a[i].x>a[j].x&&a[i].y>a[j].y&&dp[j]>oo)//+a[i].z>oo  
                       oo=dp[j];//+a[i].z;

               }  
               dp[i]=oo+a[i].z;  
             if(dp[i]>m)  
               m=dp[i];  
           }  
        d++;  
        printf("Case %d: maximum height = %d\n",d,m);  
    }

    return 0;  
}

**Big Event in HDU**

**Time Limit: 10000/5000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 35311    Accepted Submission(s): 12250**

Problem Description

Nowadays, we all know that Computer College is the biggest department in HDU. But, maybe you don't know that Computer College had ever been split into Computer College and Software College in 2002.  
The splitting is absolutely a big event in HDU! At the same time, it is a trouble thing too. All facilities must go halves. First, all facilities are assessed, and two facilities are thought to be same if they have the same value. It is assumed that there is N (0<N<1000) kinds of facilities (different value, different kinds).

Input

Input contains multiple test cases. Each test case starts with a number N (0 < N <= 50 -- the total number of different facilities). The next N lines contain an integer V (0<V<=50 --value of facility) and an integer M (0<M<=100 --corresponding number of the facilities) each. You can assume that all V are different.  
A test case starting with a negative integer terminates input and this test case is not to be processed.

Output

For each case, print one line containing two integers A and B which denote the value of Computer College and Software College will get respectively. A and B should be as equal as possible. At the same time, you should guarantee that A is not less than B.

Sample Input

2

10 1

20 1

3

10 1

20 2

30 1

-1

Sample Output

20 10

40 40

Author

 一维背包,逐个考虑每个物品带来的影响,对于第i个物品:if(f[j-v[i]]==0) f[j]=0;  
    其中,j为逆序循环,且j>=v[i]    

|  |
| --- |
|  |
| **I NEED A OFFER!**  **Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others) Total Submission(s): 24473    Accepted Submission(s): 9845**  Problem Description  Speakless 很早就想出国，现在他已经考完了所有需要的考试，准备了所有要准备的材料，于是，便需要去申请学校了。要申请国外的任何大学，你都要交纳一定的申请费用， 这可是很惊人的。Speakless没有多少钱，总共只攒了n万美元。他将在m个学校中选择若干的（当然要在他的经济承受范围内）。每个学校都有不同的申 请费用a（万美元），并且Speakless估计了他得到这个学校offer的可能性b。不同学校之间是否得到offer不会互相影响。“I NEED A OFFER”，他大叫一声。帮帮这个可怜的人吧，帮助他计算一下，他可以收到至少一份offer的最大概率。（如果Speakless选择了多个学校，得 到任意一个学校的offer都可以）。    Input  输入有若干组数据，每组数据的第一行有两个正整数n,m(0<=n<=10000,0<=m<=10000)  后面的m行，每行都有两个数据ai(整型),bi(实型)分别表示第i个学校的申请费用和可能拿到offer的概率。  输入的最后有两个0。    Output  每组数据都对应一个输出，表示Speakless可能得到至少一份offer的最大概率。用百分数表示，精确到小数点后一位。    Sample Input  10 3  4 0.1  4 0.2  5 0.3  0 0    Sample Output  44.0%  ***Hint***  You should use printf("%%") to print a '%'.      Author  Speakless    Source |

简单0-1背包,题目要求的是至少收到一份Offer的最大概率,我们得到得不到的最小概率即可,状态转移方程:f[j]=min(f[j],f[j-v[i]]\*w[i]);其中,w[i]表示得不到的概率,(1-f[j])为花费j元得到Offer的最大概率

**Doing Homework again**

**Time Limit: 1000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 11458    Accepted Submission(s): 6733**

Problem Description

Ignatius has just come back school from the 30th ACM/ICPC. Now he has a lot of homework to do. Every teacher gives him a deadline of handing in the homework. If Ignatius hands in the homework after the deadline, the teacher will reduce his score of the final test. And now we assume that doing everyone homework always takes one day. So Ignatius wants you to help him to arrange the order of doing homework to minimize the reduced score.

Input

The input contains several test cases. The first line of the input is a single integer T that is the number of test cases. T test cases follow.  
Each test case start with a positive integer N(1<=N<=1000) which indicate the number of homework.. Then 2 lines follow. The first line contains N integers that indicate the deadlines of the subjects, and the next line contains N integers that indicate the reduced scores.

Output

For each test case, you should output the smallest total reduced score, one line per test case.

Sample Input

3

3

3 3 3

10 5 1

3

1 3 1

6 2 3

7

1 4 6 4 2 4 3

3 2 1 7 6 5 4

Sample Output

0

3

5

#include<stdio.h>

#include<string.h>

#include<algorithm>

using namespace std;

const int MAXN=1010;

struct Node

{

int d,s;

}node[MAXN];

bool used[10000];

bool cmp(Node a,Node b)

{

if(a.s==b.s)

{

return a.d<b.d;

}

return a.s>b.s;

}

int main()

{

int T;

int n;

int j;

scanf("%d",&T);

while(T--)

{

scanf("%d",&n);

for(int i=0;i<n;i++) scanf("%d",&node[i].d);

for(int i=0;i<n;i++) scanf("%d",&node[i].s);

sort(node,node+n,cmp);

memset(used,false,sizeof(used));

int ans=0;

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

{

for(j=node[i].d;j>0;j--)

{

if(!used[j])

{

used[j]=true;

break;

}

}

if(j==0)

ans+=node[i].s;

}

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

}

return 0;

}

**Advanced Fruits**

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 2757    Accepted Submission(s): 1405  
Special Judge**

Problem Description

The company "21st Century Fruits" has specialized in creating new sorts of fruits by transferring genes from one fruit into the genome of another one. Most times this method doesn't work, but sometimes, in very rare cases, a new fruit emerges that tastes like a mixture between both of them.   
A big topic of discussion inside the company is "How should the new creations be called?" A mixture between an apple and a pear could be called an apple-pear, of course, but this doesn't sound very interesting. The boss finally decides to use the shortest string that contains both names of the original fruits as sub-strings as the new name. For instance, "applear" contains "apple" and "pear" (APPLEar and apPlEAR), and there is no shorter string that has the same property.   
  
A combination of a cranberry and a boysenberry would therefore be called a "boysecranberry" or a "craboysenberry", for example.   
  
Your job is to write a program that computes such a shortest name for a combination of two given fruits. Your algorithm should be efficient, otherwise it is unlikely that it will execute in the alloted time for long fruit names.

Input

Each line of the input contains two strings that represent the names of the fruits that should be combined. All names have a maximum length of 100 and only consist of alphabetic characters.  
  
Input is terminated by end of file.

Output

For each test case, output the shortest name of the resulting fruit on one line. If more than one shortest name is possible, any one is acceptable.

Sample Input

apple peach

ananas banana

pear peach

Sample Output

appleach

bananas

pearch

Source

[University of Ulm Local Contest 1999](http://acm.hdu.edu.cn/search.php?field=problem&key=University+of+Ulm+Local+Contest+1999&source=1&searchmode=source)

#include<cstdio>

#include<cstring>

char a[105],b[105];

int pre[105][105];

void out(int i,int j)

{

if(!i&&!j)return;

if(pre[i][j]==j+1)

{

out(i-1,j-1);

putchar(a[i]);

}

else if(pre[i][j]==j)

{

out(i-1,j);

putchar(a[i]);

}

else

{

out(i,j-1);

putchar(b[j]);

}

}

void LCS()

{

int n=strlen(a+1),m=strlen(b+1),i,j,dp[105][105]={0};

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

for(j=1;j<=m;j++)

if(a[i]==b[j])

{

dp[i][j]=dp[i-1][j-1]+1;

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

}

else if(dp[i][j-1]>dp[i-1][j])

{

dp[i][j]=dp[i][j-1];

pre[i][j]=j-1;

}

else

{

dp[i][j]=dp[i-1][j];

pre[i][j]=j;

}

out(n,m);

puts("");

}

int main()

{

while(scanf("%s%s",a+1,b+1)==2)

LCS();

}

**Regular Words**

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 1990    Accepted Submission(s): 771**

Problem Description

Consider words of length 3n over alphabet {A, B, C} . Denote the number of occurences of A in a word a as A(a) , analogously let the number of occurences of B be denoted as B(a), and the number of occurenced of C as C(a) .   
  
Let us call the word w regular if the following conditions are satisfied:   
  
A(w)=B(w)=C(w) ;   
if c is a prefix of w , then A(c)>= B(c) >= C(c) .   
For example, if n = 2 there are 5 regular words: AABBCC , AABCBC , ABABCC , ABACBC and ABCABC .   
  
Regular words in some sense generalize regular brackets sequences (if we consider two-letter alphabet and put similar conditions on regular words, they represent regular brackets sequences).   
  
Given n , find the number of regular words.

Input

There are mutiple cases in the input file.   
  
Each case contains n (0 <= n <= 60 ).   
  
There is an empty line after each case.

Output

Output the number of regular words of length 3n .   
  
There should be am empty line after each case.

Sample Input

2

3

Sample Output

5

42

Source

[Andrew Stankevich's Contest #10](http://acm.hdu.edu.cn/search.php?field=problem&key=Andrew+Stankevich%27s+Contest+%2310&source=1&searchmode=source)

Recommend

xhd

#include <stdio.h>

#include <string.h>

#include <algorithm>

using namespace std;

char dp[65][65][65][85];

void add(char a[],char b[],char back[])

{

int i,j,k,up,x,y,z,l;

char \*c;

if(strlen(a) > strlen(b))

l = strlen(a)+2;

else

l = strlen(b)+2;

c = (char\*)malloc(l\*sizeof(char));

i = strlen(a)-1;

j = strlen(b)-1;

k = 0;

up = 0;

while(j>=0 || i>=0)

{

if(i<0) x = '0';

else

x = a[i];

if(j<0) y = '0';

else

y = b[j];

z = x-'0'+y-'0';

if(up)

z++;

if(z>9)

{

up = 1;

z%=10;

}

else

up = 0;

c[k++] = z+'0';

i--;

j--;

}

if(up)

c[k++] = '1';

i = 0;

c[k] = '\0';

for(k-=1; k>=0; k--)

back[i++] = c[k];

back[i] = '\0';

}

int main()

{

int n,i,j,k;

for(i = 0; i<=60; i++)

for(j = 0; j<=60; j++)

for(k = 0; k<=60; k++)

strcpy(dp[i][j][k],"0");

strcpy(dp[1][0][0],"1");

strcpy(dp[1][1][0],"1");

strcpy(dp[1][1][1],"1");

for(i = 2; i<=60; i++)

{

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

{

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

{

if(i-1>=j)

add(dp[i-1][j][k],dp[i][j][k],dp[i][j][k]);

if(j-1>=k)

add(dp[i][j-1][k],dp[i][j][k],dp[i][j][k]);

if(j>=k-1)

add(dp[i][j][k],dp[i][j][k-1],dp[i][j][k]);

}

}

}

while(~scanf("%d",&n))

{

printf("%s\n\n",dp[n][n][n]);

}

return 0;

}

**Warcraft**

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 1570    Accepted Submission(s): 817**

Problem Description

Have you ever played the Warcraft?It doesn't matter whether you have played it !We will give you such an experience.There are so many Heroes in it,but you could only choose one of them.Each Hero has his own skills.When such a Skill is used ,it costs some MagicValue,but hurts the Boss at the same time.Using the skills needs intellegence,one should hurt the enemy to the most when using certain MagicValue.  
  
Now we send you to complete such a duty to kill the Boss(So cool~~).To simplify the problem:you can assume the LifeValue of the monster is 100, your LifeValue is 100,but you have also a 100 MagicValue!You can choose to use the ordinary Attack(which doesn't cost MagicValue),or a certain skill(in condition that you own this skill and the MagicValue you have at that time is no less than the skill costs),there is no free lunch so that you should pay certain MagicValue after you use one skill!But we are good enough to offer you a "ResumingCirclet"(with which you can resume the MagicValue each seconds),But you can't own more than 100 MagicValue and resuming MagicValue is always after you attack.The Boss is cruel , be careful!

Input

There are several test cases,intergers n ,t and q (0<n<=100，1<=t<=5，q>0) in the first line which mean you own n kinds of skills ,and the "ResumingCirclet" helps you resume t points of MagicValue per second and q is of course the hurt points of LifeValue the Boss attack you each time(we assume when fighting in a second the attack you show is before the Boss).Then n lines follow,each has 2 intergers ai and bi(0<ai,bi<=100).which means using i skill costs you ai MagicValue and costs the Boss bi LifeValue.The last case is n=t=q=0.

Output

Output an interger min (the minimun time you need to kill the Boss)in one line .But if you die(the LifeValue is no more than 0) ,output "My god"!

Sample Input

4 2 25

10 5

20 10

30 28

76 70

4 2 25

10 5

20 10

30 28

77 70

0 0 0

Sample Output

4

My god

***Hint***

Hint:

When fighting,you can only choose one kind of skill or just to use the ordinary attack in the whole second,the ordinary attack costs the Boss 1

points of LifeValue,the Boss can only use ordinary attack which costs a whole second at a time.Good Luck To You!

Source

[2009 Multi-University Training Contest 11 - Host by HRBEU](http://acm.hdu.edu.cn/search.php?field=problem&key=2009+Multi-University+Training+Contest+11+-+Host+by+HRBEU&source=1&searchmode=source)

#pragma comment(linker, "/STACK:1024000000,1024000000")

#include<iostream>

#include<cstdio>

#include<cstring>

#include<string>

#include<algorithm>

#include<queue>

#include<set>

#include<stack>

#include<cmath>

#include<map>

#include<stdlib.h>

#include<cctype>

#define mem(a,x) memset(a,x,sizeof(a))

#define esp 1e-8

using namespace std;

typedef long long ll;

const int N = 100;

int a[N+5],b[N+5];

int dp[N+5][N+5];

int main()

{

int n,t,q;

while (scanf("%d %d %d",&n,&t,&q) == 3&&(n||t||q))

{

for (int i = 1;i <= n;++i) scanf("%d %d",a+i,b+i);

a[0] = 0,b[0] = 1;//普通攻击

mem(dp,-1);int dt = 100/q;

if (100%q) ++dt;

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

{

if (a[i]<=100)//技能花费

{

dp[1][100-a[i]] = b[i];

}

}

for (int i = 2;i <= dt;++i)

{

for (int j = 0;j <= 100;++j)

{

for (int k = 0;k <= n;++k)

{

if (dp[i-1][j] == -1) continue;//转移的前状态必须是存在的才能转移

if (min(100,j+t) >= a[k])//该技能可以使用

{ //那么进行决策

dp[i][min(j-a[k]+t,100)] = max(dp[i][min(j-a[k]+t,100)],dp[i-1][j] + b[k]);

}

}

}

}

bool fd = 0;

for (int i = 1;i <= dt;++i)//找最小时间

{

for (int j = 0;j <= 100;++j)

{

if (dp[i][j] >= 100)

{

fd = 1;

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

break;

}

}

if (fd) break;

}

if (!fd) puts("My god");

}

return 0;

}

**Fast Food**

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 2753    Accepted Submission(s): 1173**

Problem Description

The fastfood chain McBurger owns several restaurants along a highway. Recently, they have decided to build several depots along the highway, each one located at a restaurant and supplying several of the restaurants with the needed ingredients. Naturally, these depots should be placed so that the average distance between a restaurant and its assigned depot is minimized. You are to write a program that computes the optimal positions and assignments of the depots.   
  
To make this more precise, the management of McBurger has issued the following specification: You will be given the positions of n restaurants along the highway as n integers d1 < d2 < ... < dn (these are the distances measured from the company's headquarter, which happens to be at the same highway). Furthermore, a number k (k <= n) will be given, the number of depots to be built.   
  
The k depots will be built at the locations of k different restaurants. Each restaurant will be assigned to the closest depot, from which it will then receive its supplies. To minimize shipping costs, the total distance sum, defined as

http://acm.hdu.edu.cn/data/images/1227-1.gif

must be as small as possible.  
  
Write a program that computes the positions of the k depots, such that the total distance sum is minimized.

Input

The input file contains several descriptions of fastfood chains. Each description starts with a line containing the two integers n and k. n and k will satisfy 1 <= n <= 200, 1 <= k <= 30, k <= n. Following this will n lines containing one integer each, giving the positions di of the restaurants, ordered increasingly.  
  
The input file will end with a case starting with n = k = 0. This case should not be processed.

Output

For each chain, first output the number of the chain. Then output a line containing the total distance sum.   
  
Output a blank line after each test case.

Sample Input

6 3

5

6

12

19

20

27

0 0

Sample Output

Chain 1

Total distance sum = 8

Source

[Southwestern Europe 1998](http://acm.hdu.edu.cn/search.php?field=problem&key=Southwestern+Europe+1998&source=1&searchmode=source)

Recommend

#include<stdio.h>

#include<string.h>

const int maxn=220;

const int INF=99999999;

int dis[maxn],dp[maxn][maxn],cost[maxn][maxn];

int abs(int x){

return x<0?-x:x;

}

int min(int a,int b){

return a<b?a:b;

}

int main(){

//freopen("input.txt","r",stdin);

int n,k;

int cases=0;

while(scanf("%d%d",&n,&k)){

int i,j,m;

if(n==0 && k==0)

break;

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

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

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

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

cost[i][j]=0;

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

cost[i][j]+=abs(dis[m]-dis[(i+j)/2]);

}

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

dp[1][i]=cost[1][i];

for(i=2;i<=k;i++)

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

dp[i][j]=INF;

for(m=i-1;m<=j-1;m++)

dp[i][j]=min(dp[i][j],dp[i-1][m]+cost[m+1][j]);

}

printf("Chain %d\nTotal distance sum = %d\n\n",++cases,dp[k][n]);

}

return 0;

}

|  |
| --- |
|  |
| **Zipper**  **Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others) Total Submission(s): 9420    Accepted Submission(s): 3341**  Problem Description  Given three strings, you are to determine whether the third string can be formed by combining the characters in the first two strings. The first two strings can be mixed arbitrarily, but each must stay in its original order.  For example, consider forming "tcraete" from "cat" and "tree":  String A: cat String B: tree String C: tcraete   As you can see, we can form the third string by alternating characters from the two strings. As a second example, consider forming "catrtee" from "cat" and "tree":  String A: cat String B: tree String C: catrtee   Finally, notice that it is impossible to form "cttaree" from "cat" and "tree".    Input  The first line of input contains a single positive integer from 1 through 1000. It represents the number of data sets to follow. The processing for each data set is identical. The data sets appear on the following lines, one data set per line.  For each data set, the line of input consists of three strings, separated by a single space. All strings are composed of upper and lower case letters only. The length of the third string is always the sum of the lengths of the first two strings. The first two strings will have lengths between 1 and 200 characters, inclusive.    Output  For each data set, print:  Data set n: yes  if the third string can be formed from the first two, or  Data set n: no  if it cannot. Of course n should be replaced by the data set number. See the sample output below for an example.    Sample Input  3  cat tree tcraete  cat tree catrtee  cat tree cttaree    Sample Output  Data set 1: yes  Data set 2: yes  Data set 3: no |

#include <bits/stdc++.h>

using namespace std;

char s1[210], s2[210], s3[420];

bool dp[1010][1010];

int main()

{

int tp1, tp2, tp3, T, k, i, j;

int len1, len2;

k = 1;

scanf("%d\n", &T);

while(T--){

scanf("%s %s %s", s1+1, s2+1, s3+1);

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

dp[0][0] = 1;

len1 = strlen(s1+1), len2 = strlen(s2+1);

for(i = 0;i <= len1;i++)

for(j = 0;j <= len2;j++){

if(i > 0&&(dp[i-1][j]&(s1[i]==s3[i+j]))) dp[i][j] = 1;

else if(j > 0&&(dp[i][j-1]&(s2[j]==s3[i+j]))) dp[i][j] = 1;

else if(i>0||j>0)dp[i][j] = 0;

//printf("%d %d\n", (dp[i][j-1]&(s2[j]==s3[i+j])), (dp[i-1][j]&(s1[i]==s3[i+j])));

}

printf("Data set %d: %s\n", k++, dp[len1][len2]?"yes":"no");

}

}

**Pearls**

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 2323    Accepted Submission(s): 1117**

Problem Description

In Pearlania everybody is fond of pearls. One company, called The Royal Pearl, produces a lot of jewelry with pearls in it. The Royal Pearl has its name because it delivers to the royal family of Pearlania. But it also produces bracelets and necklaces for ordinary people. Of course the quality of the pearls for these people is much lower then the quality of pearls for the royal family. In Pearlania pearls are separated into 100 different quality classes. A quality class is identified by the price for one single pearl in that quality class. This price is unique for that quality class and the price is always higher then the price for a pearl in a lower quality class.  
  
Every month the stock manager of The Royal Pearl prepares a list with the number of pearls needed in each quality class. The pearls are bought on the local pearl market. Each quality class has its own price per pearl, but for every complete deal in a certain quality class one has to pay an extra amount of money equal to ten pearls in that class. This is to prevent tourists from buying just one pearl.  
  
Also The Royal Pearl is suffering from the slow-down of the global economy. Therefore the company needs to be more efficient. The CFO (chief financial officer) has discovered that he can sometimes save money by buying pearls in a higher quality class than is actually needed. No customer will blame The Royal Pearl for putting better pearls in the bracelets, as long as the prices remain the same.  
  
For example 5 pearls are needed in the 10 Euro category and 100 pearls are needed in the 20 Euro category. That will normally cost: (5+10)\*10 + (100+10)\*20 = 2350 Euro.  
  
Buying all 105 pearls in the 20 Euro category only costs: (5+100+10)\*20 = 2300 Euro.  
  
The problem is that it requires a lot of computing work before the CFO knows how many pearls can best be bought in a higher quality class. You are asked to help The Royal Pearl with a computer program.  
  
Given a list with the number of pearls and the price per pearl in different quality classes, give the lowest possible price needed to buy everything on the list. Pearls can be bought in the requested, or in a higher quality class, but not in a lower one.

Input

The first line of the input contains the number of test cases. Each test case starts with a line containing the number of categories c (1 <= c <= 100). Then, c lines follow, each with two numbers ai and pi. The first of these numbers is the number of pearls ai needed in a class (1 <= ai <= 1000). The second number is the price per pearl pi in that class (1 <= pi <= 1000). The qualities of the classes (and so the prices) are given in ascending order. All numbers in the input are integers.

Output

For each test case a single line containing a single number: the lowest possible price needed to buy everything on the list.

Sample Input

2

2

100 1

100 2

3

1 10

1 11

100 12

Sample Output

330

1344

#include<stdio.h>

#include<string.h>

#define inf 210000000

int main()

{

int T,i,j,a[105],dp[105],min,n,t,p[105];

scanf("%d",&T);

while (T--)

{

scanf("%d",&n);

a[0]=0;

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

{

scanf("%d%d",&t,&p[i]);

a[i]=a[i-1]+t;

}

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

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

{

min=inf;

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

{

if (dp[j]+(a[i]-a[j]+10)\*p[i]<min) min=dp[j]+(a[i]-a[j]+10)\*p[i];

}

dp[i]=min;

}

printf("%d/n",dp[n]);

}

return 0;

}

**重温世界杯**

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 6270    Accepted Submission(s): 2207**

Problem Description

世界杯结束了,意大利人连本带利的收回了法国人6年前欠他们的债,捧起了大力神杯,成就了4星意大利.  
世界杯虽然结束了,但是这界世界杯给我们还是留下许多值得回忆的东西.比如我们听到了黄名嘴的3分钟激情解说,我们懂得了原来可以向同一个人出示3张黄牌，我们还看到了齐达内的头不仅能顶球还能顶人…………  
介 于有这么多的精彩,xhd决定重温德国世界杯,当然只是去各个承办世界杯比赛的城市走走看看.但是这需要一大比钱,幸运的是xhd对世界杯的热爱之情打动 了德国世界杯组委会,他们将提供xhd在中国杭州和德国任意世界杯承办城市的往返机票,并说服了这些城市在xhd到达这座城市时为他提供一笔生活费以便他 在那里参观时用,当参观完时剩余的钱也将留给xhd,但当生活费不够时他们将强行结束xhd的这次德国之行,除了这个,他们还有一个条件,xhd只能根据 他们所给的路线参观.比如有3座城市a,b,c,他们给定了a-b-c-a的路线,那么xhd只有3种参观顺序abc,bca,cab.由于各个城市所提 供的生活费和在那里的花费都不同,这使xhd很头痛,还好我们事先知道了这笔生活费和花费.请问xhd最多能顺利参观几座城市?

Input

每 组输入数据分两行,第一行是一个正整数n(1<=n<=100000),表示有n座城市.接下来的一行按照给定的路线顺序的输出这n个城市的 生活费和花费,w1,l1,w2,l2,……,wn,ln,其中wi,li分别表示第i个城市的生活费和花费,并且它们都是正整数.

Output

对应每组数据输出最多能参观的城市数.

Sample Input

3

3 2 3 4 2 2

3

3 2 3 4 2 3

Sample Output

3

2

Author

xhd

Source

#include<iostream>

#include<cstdio>

#define N 200010

using namespace std;

int a[N],ans[N];

int main()

{

int n,i,t1,t2;

while(scanf("%d",&n)==1)

{

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

{

scanf("%d%d",&t1,&t2);

a[i]=a[i+n]=t1-t2;

ans[i]=ans[i+n]=1;

}

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

{

if(a[i]+a[i-1]>=0&&a[i-1]>=0)

{

a[i]=a[i]+a[i-1];

ans[i]+=ans[i-1];

if(ans[i]==n) break;

}

}

int max\_ans=0;

for(i=0;i<2\*n;i++)

if(max\_ans<ans[i])

max\_ans=ans[i];

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

}

return 0;

}

/\*

5

3 1 1 2 1 4 4 2 1 2

\*/