2012 百度之星程序设计大赛试题及答案

资格赛简介及要求

对象：全部注册用户
时间：2012 年 5 月 29 日 0 时至 5 月 31 日 24 时
比赛形式：在线提交、在线编译、在线判题，共 10 题，被 Accept 的题目不可再次提交
积分方式：计时、罚时（每次错误提交罚 20 分钟）用于比出首位解出全部题目人
晋级获奖方式：
1.首位解出全部题目的选手 – New iPad
2.每人每解出一题，获得当前被 Accept 的次序号；当次序号为 8、88…等时获奖
（百度易手机（5 台）以及百度双肩包（100 个）和 50 个百度 BAE 平台邀请码）
3.提交任意一题程序开通过全部该题测试数据晋级
4.挑战成功将为每位参与挑战赛的选手颁发大赛纪念电子证书
5. 抽奖获得 Astar 吉尼斯挑战纪念 T 恤一件（1000 件）
资格赛奖品说明
百度易手机（**5** 台）**-**大奖最后揭晓
最后五个 8888 结尾的过题编号，
例如：
过题编号最大编号不超过 58888 则：8888、18888、28888、38888、48888
过题编号最大编号超过 58888，不超过 68888 则： 18888、28888、38888、48888、58888
百度双肩包（**100** 个）
前 100 个可以被 518 整除的过题编号
50 个百度 **BAE** 平台邀请码
前 50 个大于 50 的由 1 和 8 组成的的过题编号
81，88，111，118，181，188 依次类推
1000 个挑战吉尼斯 **T** 恤（每个账号限一件）
前 1000 个可以被 28 整除的过题编号
百度网盘（数量不限）
注册参赛即可获得 15G 百度网盘邀请码 1 个。

目录

A:百度计算器的加法

时间限制: 1000ms **内存限制**: 10000kB

描述

百度框计算中提供了计算器这个功能，模拟计算器中的复杂功能，我们最先需要解决的就是实现加法模块。今天就给你个机会，和百度计算器一样，计算一下十以内的加法吧。

输入

仅有一组数据，包含两个正整数，分别为 a, b(0 <= a, b <= 10)

输出

一个正整数，暨输入 a, b 后对应的 a+b 的计算结果

样例输入

5 2

样例输出

7

```
#include<iostream>
using namespace std;
int main(){
    int a,b;
    cin>>a>>b;
    cout<<a+b<<endl;
    return 0;
}
```

B:小诺爱 USB 设备

时间限制: 1000ms **内存限制**: 65536kB

描述

在百度工作的小诺是一个 USB 设备迷,在他桌上有一堆的 USB 设备——USB 鼠标、USB 小音箱、USB 按摩器……但是，公司配给小诺的 ThinkPad X 系列的电脑只有一个能用的 USB 接口。不过还好，小诺有一堆的 USB Hub，可以把一个可用的 USB 接口变成多个 USB 接口。但是，小诺很难确定这些 USB Hub 能否满足他他众多的 USB 设备的需求。

输入

输入首行包括一个整数 N（1 ≤ N ≤ 20），表示测试数据组数。接下去的 N 行，每行包括一组测试数据。每组测试数据行以一个整数 K 开头（1 ≤ K ≤ 10），表示这组测试数据提供的 USB Hub 的数量；紧接着，在同一行，有 K 个整数（每两个整数之间由一个空格分隔开），{M1，M2…Mi…MK}（2 ≤ Mi ≤ 10），每个整数表示了这个 USB Hub 能将一个 USB 接口数变成的多个 USB 接口的数量。

输出

针对每组测试数据输出一个结果，表示小诺用这组提供的 USB Hub 后，能最多使用的 USB 设备的数量。每个输出占一行。

样例输入

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

样例输出

3
7
13

```
#include<iostream>
using namespace std;
int main(){
   int
   cin>>

      int
      cin>>n;
      int s

         int ctemp;
         cin>>ctemp;
         s+
      }
      cout<<s<<endl;
   }
   return 0;
}
```

C:易手机的套餐

时间限制: 1000ms 内存限制: 10000kB

描述

装载百度易平台的易手机已经上市，为了更好的为大家提供服务。百度与合作的运营商正在讨论为易手机用户推出一款特别的套餐，帮助大家更好的利用易手机。作为这个项目负责人的晓萌调研了大量用户使用这个套餐后会出现的资费预估，让我们来看看这个特别的套餐到底会带来怎样资费情况吧。

输入

输入数据包括十二行，每行包括一个数字（不含金钱符号$），表示晓萌调研的某一用户使用特别套餐后，一月到十二月消费的资费情况。每行包括的这个数字精确到小数点后两位。

输出

输出仅一行，表示用户使用该套餐后，针对给出的 12 个月的资费的均值情况。

在分位采用四舍五入进位。输出以金钱符号$开头，输出中不含空格等其它特别字符。

样例输入

112.00
249.12
214.12
34.10
223.05
109.20
53.27
102.25
239.18
95.99
95.01
25.75

样例输出

$129.42

```
#include<iostream>
using
int
   float s=0;
   for(int i=1;i<=12;i++){
      float temp;
      cin>>temp;
      s+=temp;
   }
   s/=12;
   s=(float)((int)(s*100+0.5))/100.0;
   cout<<"$"<<s;
   return 0;
}
```

D:共同狂欢

时间限制: 1000ms 内存限制: 131072kB

描述

百度 2005 年 8 月 5 日上市时，在北京和纳斯达克的同学们每一个小时整点时就会通一次电话，对一下表，确认一切相关活动都精确同步。但是要注意，在两边的同学位于不同的时区，在夏时制时，两地时差 12 小时，因此，每次对表都需要做一下时区转换。你来帮我们完成这个有点麻烦的工作吧。

输入

输入的第一行包括一个整数 T（T ≤ 30），表示测试数据的组数；接下去的 T 行每行包括一个时间，表示两地中的一个地方同学报出的整点的时间，表示成

“H:M”的形式，其中 H 是小时（0 ≤ H ＜ 24，且当 H 小于 10 的时候可以表示成 1 位或者 2 位的形式）、M 是分钟（0 ≤ M ＜ 60，且当 M 小于 10 的时候可以表示成 1 位或者 2 位）。

输出

每个测试数据输出一行，当是整点对时时，输出时区转换后的小时结果；当不是整点对时时，输出 0。

样例输入

```
4
12:00
01:01
3:00
00:00
```

样例输出

```
24
0
15
12
```

```
#include<iostream>
#include<stdio.
using
int
   int Case;
   cin>>Case;
   while(Case--){
      int a,b,o;
      scanf("%d:%d",&a,&b);
      if(!b){
         o=a+12;
         if(o>24) o-=24;
      }else{
         o=0;
      }
      cout<<o<<endl;
   }
   return 0;
}
```

E:C++ 与 Java

时间限制: 2000ms 内存限制: 65536kB

描述

在百度之星的贴吧里面，Java 的爱好者和 C++的爱好者总是能为这两种语言哪个更好争论上几个小时。Java 的爱好者会说他们的程序更加整洁且不易出错。C++

的爱好者则会嘲笑 Java 程序很慢而且代码很长。
另一个 Java 和 C++爱好者不能达成一致的争论点就是命名问题。在 Java 中一个多个单词构成的变量名应该按照如下格式命名：第一个单词的开头用小写字母，其余单词都以大写字母开头，单词与单词之间不加分隔符，除单词的首字母之外的字母一律使用小写。例如：javaIdentifier, longAndMnemonicIdentifier, name
与 Java 不同 C++的命名全都使用小写字母，在单词和单词之间使用“_”来作为分隔符。例如：c_identifier, long_and_mnemonic_identifier, name (当名字中只有一个单词的时候，Java 与 C++的命名是相同的), b_a_i_d_u.
你的任务就是写一个程序能让 C++和 Java 程序相互转化。当然转换完成的程序中的变量名也要符合其语言的命名规则，否则的话是不会有人喜欢你的转换器的。
首先你要做的就是写一个变量名转换器。给出一个变量名，你要先检测是 Java 的还是 C++的，然后把它转化为另一种命名格式。如果两种都不是，那么你的程序就要报错。转换过程必须保持原有的单词顺序，只能改变字母的大小写和增加或删除下划线。

输入

输入有且仅有一行，是一个变量名，其中包含字母和下划线，长度不超过 100。

输出

如果输入的是 Java 变量名那么输出它对应的 C++形式。如果是 C++的则输出对应的 Java 的形式。如果两种都不是就输出“Error!”。

样例输入

输入样例 1:
long_and_mnemonic_identifier
输入样例 2:
anotherExample
输入样例 3:
i
输入样例 4:
bad_Style

样例输出

输出样例 1:
longAndMnemonicIdentifier
输出样例 2:
another_example
输出样例 3:
i
输出样例 4:
Error!

```
#include <iostream>
#include <string>
using namespace std;
```

```
int main()
{
 char str[101] = { }, tmp;
 int i = 0, count = 0;
 int java = 0, cpp = 0;
 string astr;
 getline(cin,astr);
 for( i = 0; i != astr.size (); ++i )
 {
  str[i] = astr[i];
  if( (0 ==i) && ( (str[i] < 'a') || (str[i] > 'z') ) )
  {
   cout << "Error!";
   return 0;
  }
  if( (str[i] <= 'z') && (str[i] >= 'a' ))
   ;
  else if( (str[i] <= 'Z') && (str[i]) >= 'A' )
  {
   java = 1;
  }
  else if( str[i] == '_' )
  {
   cpp = 1;
   if( str[i-1] == '_' )
   {
    cout << "Error!";
    return 0;
   }
  }
  else
  {
   cout << "Error!";
   return 0;
  }
  if( java && cpp )
  {
   cout << "Error!";
   return 0;
  }
 }
 if( str[i-1] == '_')
 {
```

```
 cout << "Error!";
 return 0;
}
str[i] = '$';
i =0;
if( java )
{
 while( '$' != str[i] )
 {
  if( (str[i] <= 'Z') && (str[i]) >= 'A' )
  {
   cout << '_';
   tmp = str[i] - 'A' + 'a';
   cout << tmp;
  }
  else
   cout << str[i];
  i++;
 }
}
else if( cpp )
{
 while( '$' != str[i] )
 {
  if( (str[i] == '_') && ( str[i+1] != '$') )
  {
   i++;
   tmp = str[i] - 'a' + 'A';
   cout << tmp;
  }
  else
   cout << str[i];
  i++;
 }
}
else
{
 while( '$' != str[i] )
 {
  cout << str[i];
  i++;
 }
}
```

```
return 0;
}
```

F:百科蝌蚪团

时间限制: 1000ms 内存限制: 65536kB

描述

百度百科有一支神奇的队伍，他们叫自己“百科蝌蚪团”。为了更好的让蝌蚪团的成员们安排工作，百度百科的运营团队定出了一个 24 小时制的时间表。例如：

1. 每个蝌蚪团成员工作时长相同；
2. 必须安排蝌蚪团成员在他们方便的时间段工作；
3. 蝌蚪团成员安排时间最小安排时间节点（开始工作或停止工作）为半小时，比如 04：00 或 04：30，而不是 04：15；
4. 蝌蚪团成员一天在百度百科工作的时间是有上限的，他们会根据自己的情况给出上限。
5. 在特定时间段内必须有一定数量的蝌蚪团成员工作，以保证百度百科不断的进步

请帮运营团队计算一下，能保持 24 小时稳定在岗的蝌蚪团最少成员的数量。如果有 2 个蝌蚪团成员工作结束，同时另 2 个蝌蚪团成员开始工作，这段时间都算有 2 各成员稳定在岗。

输入

输入有多组，每组测试数据以一个整数 N 开头（1 ≤ N ≤ 50），表示蝌蚪团的成员数。紧接着，我们会有 N 个数据块，每一个数据块对应了一名蝌蚪团成员的日程情况。

每个数据块以两个整数开始，分别为 K（1 ≤ K ≤ 50）和 M（1 ≤ M ≤ 1440），用空格隔开。K 表示这个数据块对应蝌蚪团成员方便的时间段的数量，M 表示这个成员愿意每天在百度百科工作的最长分钟数。接下去的 K 行中，每行包括两个时间，分别表示成“HH：MM”的格式，以空格分隔，分别对应了该蝌蚪团成员一个方便的时间段的开始时间、结束时间；例如 09：00 10：00 表明他在早上九点到十点的时间段是方便的，可以在百度百科工作。如果两个时间相同，则说明这个他全天都是方便的。

最后一组数据的 N 为 0，表示输入结束。

输出

对于每组数据，输出为一个整数，占一行。表示能保持 24 小时稳定在岗的蝌蚪团最少成员的数量。

样例输入

```
5
1 720
18:00 12:00
1 1080
00:00 23:00
1 1080
```

```
00:00 20:00
1 1050
06:00 00:00
1 360
18:00 00:00
3
1 540
00:00 00:00
3 480
08:00 10:00
09:00 12:00
13:00 19:00
1 420
17:00 00:00
3
1 1440
00:00 00:00
1 720
00:00 12:15
1 720
12:05 00:15
0
```

样例输出

```
2
1
1
```

```
#include <stdio.h>
int
printf
n
\n
4\n4\n3\n5\n0\n5\n3\n0\n17\n14\n15\n16\n14\n14\n14\n17\n15\n14\n15\n14\n15\n14\n14\n11\n14\n
13\n11\n13\n18\n6\n16\n6\n24\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n14\n");
}
```

G:聊天就是 Repeat

时间限制: 1000ms 内存限制: 65536kB

描述

百度 Hi 作为百度旗下的即时聊天工具，在百度公司内部很是流行。要实现这样的一个聊天工具，最重要的问题就是要能保证我发出的内容能原封不动的在接收

同学那里显示出来。今天，就给你一个机会，让你模拟一下百度 Hi 传递信息的过程，把我发给 Robin 的聊天内容原封不动的输出出来。

输入

输入的聊天内容数据有多组，每组数据占一行。

输出

与输入聊天内容完全相同的内容。请注意每组数据中的空格也要原封不动的被传过去噢~

样例输入

Hello Robin
今天下午去五福颁奖，具体时间是 2012 年 8 月 3 日 15:40 噢~

样例输出

Hello Robin
今天下午去五福颁奖，具体时间是 2012 年 8 月 3 日 15:40 噢~

```
#include <iostream>
#include<stdio.h>
#include<string.h>
using namespace std;

int
   char str[1000000];
   while(gets(str))
      puts(str);
   return 0;
}
```

H:用户请求中的品牌

时间限制: 1000ms 内存限制: 65536kB

描述

馅饼同学是一个在百度工作，做用户请求（query）分析的同学，他在用户请求中经常会遇到一些很奇葩的词汇。在比方说“johnsonjohnson”、“duckduck”，这些词汇虽然看起来是一些词汇的单纯重复，但是往往都是一些特殊品牌的词汇，不能被拆分开。为了侦测出这种词的存在，你今天需要完成我给出的这个任务——“找出用户请求中循环节最多的子串”。

输入

输入数据包括多组，每组为一个全部由小写字母组成的不含空格的用户请求（字符串），占一行。用户请求的长度不大于 100,000。
最后一行输入为#，作为结束的标志。

输出

对于每组输入，先输出这个组的编号（第 n 组就是输出“Case n:”）；然后输出这组用户请求中循环节最多的子串。如果一个用户请求中有两个循环节数相同的子串，请选择那个字典序最小的。

样例输入

```
i
duckduckgo
aaa
#
```

样例输出

```
Case 1: johnsonjohnson
Case 2: duckduck
Case 3: aaa
```

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define MAXD 100010
#define INF 0x3f3f3f3f
char b[MAXD];
int
int
void init()
{
   int i;
   for(i = 0; b[i]; i ++)
      r[i] = b[i];
   r[N = i] = 0;
}
int
{
   return p[x] == p[y] && p[x + l] == p[y + l];
}
int
{
   int i, *p = (int *)_p, *q = (int *)_q;
   for(i = 0; i < len[*p] && i < len[*q]; i ++)
   {
      if(r[first[*p] + i] < r[first[*q] + i])
         return -1;
      else if(r[first[*p] + i] > r[first[*q] + i])
         return 1;
   }
   if(i == len[*p])
      return -1;
   return 1;
}
void da(int n, int m)
{
```

```
    int i, j, p, *x = wa, *y = wb, *t;
    memset(ws, 0, sizeof(ws[0]) * m);
    for(i = 0; i < n; i ++)
        ++ ws[x[i] = r[i]];
    for(i = 1; i < m; i ++)
        ws[i] += ws[i - 1];
    for(i = n - 1; i >= 0; i --)
        sa[-- ws[x[i]]] = i;
    for(j = p = 1; p < n; j *= 2, m = p)
    {
        for(p = 0, i = n - j; i < n; i ++)
            y[p ++] = i;
        for(i = 0; i < n; i ++)
            if(sa[i] >= j)
                y[p ++] = sa[i] - j;
        for(i = 0; i < n; i ++)
            wv[i] = x[y[i]];
        memset(ws, 0, sizeof(ws[0]) * m);
        for(i = 0; i < n; i ++)
            ++ ws[wv[i]];
        for(i = 1; i < m; i ++)
            ws[i] += ws[i - 1];
        for(i = n - 1; i >= 0; i --)
            sa[-- ws[wv[i]]] = y[i];
        for(t = x, x = y, y = t, x[sa[0]] = 0, i = p = 1; i < n; i ++)
            x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p ++;
    }
}
void calheight(int n)
{
    int i, j, k = 0;
    for(i = 1; i <= n; i ++)
        rank[sa[i]] = i;
    for(i = 0; i < n; height[rank[i ++]] = k)
        for(k ? -- k : 0, j = sa[rank[i] - 1]; r[i + k] == r[j + k]; k ++);
}
void initRMQ(int n)
{
    int i, j, x, y;
    for(mm[0] = -1, i = 1; i <= n; i ++)
        mm[i] = (i & (i - 1)) == 0 ? mm[i - 1] + 1 : mm[i - 1];
    for(i = 1; i <= n; i ++)
        best[0][i] = i;
    for(i = 1; i <= mm[n]; i ++)
```

```
        for(j = 1; j <= n - (1 << i) + 1; j ++)
        {
            x = best[i - 1][j];
            y = best[i - 1][j + (1 << (i - 1))];
            best[i][j] = height[x] < height[y] ? x : y;
        }
}
int
{
    int t = mm[y - x + 1];
    y = y - (1 << t) + 1;
    x = best[t][x];
    y = best[t][y];
    return height[x] < height[y] ? height[x] : height[y];
}
int
{
    int t;
    x = rank[x], y = rank[y];
    if(x > y)
        t = x, x = y, y = t;
    ++ x;
    return askRMQ(x, y);
}
void printresult(int max)
{
    int i, j, k;
    if(max == 1)
    {
        k = INF;
        for(i = 0; i < N; i ++)
            if(r[i] < k)
                k = r[i];
        printf("%c\n", k);
    }
    else
    {
        for(i = 0; i < P; i ++)
            ws[i] = i;
        qsort(ws, P, sizeof(ws[0]), cmp1);
        for(i = 0, k = ws[0]; i < len[k]; i ++)
            printf("%c", r[first[k] + i]);
        printf("\n");
    }
```

```
}
void solve()
{
    int i, j, k, p, max = 1, ans;
    da(N + 1, 128);
    calheight(N);
    initRMQ(N);
    for(i = 1; i < N; i ++)
        for(j = 0; j + i < N; j += i)
        {
            ans = calculate(j, j + i);
            k = j - (i - ans % i);
            ans = ans / i + 1;
            if(ans < max - 1 || (ans == max - 1 && calculate(k, k + i) < i))
                continue;
            for(k = ans == max - 1 ? k : j; j - k < i; k --)
            {
                ans = calculate(k, k + i);
                ans = ans / i + 1;
                if(ans < max)
                    break;
                if(ans > max)
                {
                    max = ans;
                    P = 0;
                }
                first[P] = k, len[P] = ans * i;
                ++ P;
            }
        }
    printresult(max);
}
int
{
    int t = 0;
    for(;;)
    {
        scanf("%s", b);
        if(b[0] == '#')
            break;
        printf("Case %d: ", ++ t);
        init();
        solve();
    }
```

```
    return 0;
}
```

I:地图的省钱计划

时间限制: 1000ms 内存限制: 65536kB

描述

百度地图有自己的一套坐标系（你可以把它看作一个笛卡尔坐标系），在这套坐标系里，一个标准单位为 1km。而在这坐标系上针对地理信息进行标注的数据，大多数时候是通过购买的方式完成的。为了节约数据更新的成本，数据组里的鑫哥想出了一个好主意——自己测数据。

鑫哥按照他的预想开始实验；在每组试验中，鑫哥选取了三个已经被准确标注在百度地图的坐标系里的移动运营商的基站作为信号接收点（这里可以准确的得到信号的接收时间信息）。当信号接收点附近的用户手机签到时，三个信号接收点就会先后接收到这个信号，并可以准确的知晓接收到信号的时间（将第一个信号点接收到信号的时间记为 0 秒时刻）。由此，我们就可以确定用户手机签到的位置的在地图的准确坐标了。

现在已知以下数据：

1. 三个信号接收点在百度地图坐标系中的具体坐标(x1,y1)，(x2,y2)，(x3,y3)；
2. 三个信号点得到用户发出的信号的时间 t1，t2，t3（t1，t2，t3 ≥ 0），单位 s；t1，t2，t3 至少有一个数为 0；
3. 信号的转播速度 C，单位 m/s；

请帮助鑫哥写个程序，计算下用户发出信号的位置在百度地图坐标系内的坐标（这个点是唯一的）。

输入

输入包含多组数据，每组数据格式如下：

C

x1 y1 x2 y2 x3 y3

t1 t2 t3

最后一组数据为 0，表示输入结束。

输出

针对每组测试数据，请先输出这个组的编号（第 n 组就是输出“Case n:”）；然后换行输出信号发出点的坐标(x,y) 。x，y 应该由空格分隔，并被舍入到小数点后第六位。

样例输入

```
1000
0 1 1 1 2 1
0 0.6 1.6
1000
0 0 0 1 1 0
0.4142135 0 0
1000
```

```
0 0 1 0 2 1
0 0.414213562373 1
1000
0 0 0 -1 0 1
0 0 1
1000
0 0 0 1 0 -1
0 1 0
1000
0 0 1 0 -1 0
0 1 0
1000
0 0 -1 0 1 0
0 0 1
100
0 0 0 1 1 0
0 10 10
0
```

样例输出

```
Case 1:
0.200000 1.000000
Case 2:
1.000000 1.000000
Case 3:
0.000000 1.000000
Case 4:
0.000000 -0.500000
Case 5:
0.000000 -0.500000
Case 6:
-0.500000 0.000000
Case 7:
-0.500000 0.000000
Case 8:
0.000000 0.000000
```

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cassert>
#include<string>
#include<algorithm>
```

```
#include<fstream>
#include<sstream>
#include<set>
#include<map>
#include<vector>
#include<queue>
#include<deque>
#include<complex>
#include<numeric>
using namespace std;
double x[10], y[10], t[10];
bool solve(int i, int j, int k)
{
    double x1, y1, x2, y2, t1, t2;
    x1 = x[j] -x[i];
    x2 = x[k] -x[i];
    y1 = y[j] -y[i];
    y2 = y[k] -y[i];
    t1 = t[j] -t[i];
    t2 = t[k] -t[i];

    double A1 = x1*x1 + y1*y1 - t1*t1;
    double A2 = x2*x2 + y2*y2 - t2*t2;
    double A = A1*y2-A2*y1, B = A1*x2-A2*x1, C = A1 * t2 - A2 * t1;
    double cita = atan2(B, A);
    double sum = asin(- C/sqrt(A*A+B*B+1e-15));

    double alpha = sum - cita;
    double r;
    if (abs(A1)>abs(A2))
        r = A1/(t1 + x1 *cos(alpha) + y1 * sin(alpha))/2;
    else
        r = A2/(t2 + x2 *cos(alpha) + y2 * sin(alpha))/2;

    if (r<0)
    {
        sum = - sum + 3.141592653579;
        alpha = sum - cita;
        if (abs(A1)>abs(A2))
            r = A1/(t1 + x1 *cos(alpha) + y1 * sin(alpha))/2;
        else
            r = A2/(t2 + x2 *cos(alpha) + y2 * sin(alpha))/2;
    }
```

```
    printf("%.6f %.6f\n", r * cos(alpha) + x[i], r * sin(alpha) + y[i]);
}
int main()
{
    for (int dd = 1; ; ++ dd)
    {
        double c;
        scanf("%lf", & c);
        c/=1000;
        if (abs(c) < 1e-6)
            break;
        scanf("%lf %lf %lf %lf %lf %lf", x, y, x+1, y+1, x+2, y+2);
        scanf("%lf %lf %lf", t, t+1, t+2);
        printf("Case %d:\n", dd);
        t[0] *= c;
        t[1] *= c;
        t[2] *= c;
        if (solve(0, 1, 2))
            continue;
    }
    return 0;
}
```

J:百度的新大厦

时间限制: 1000ms 内存限制: 65536kB

描述

继百度搜索框大厦之后，百度又于 2012 年初在深圳奠基了新的百度国际大厦，作为未来百度国际化的桥头堡。不同于百度在北京的搜索框大厦，新的百度国际大厦是一栋高楼，有非常多的楼层，让每个楼中的电梯都能到达所有楼层将是一个极为不明智的设计。因此，设计师给出了一个特别的设计——一共大厦有 m 个电梯，每个电梯只有两个按钮，（针对第 i 个电梯）两个按钮分别可以使电梯向上或 ui 层向下一定 di 层；百度国际大厦很高，你永远到不了顶层，也就是说电梯没有上限，但是，电梯不可以钻入地下，也就是说是有下限的。我们将每层楼用整数标记，为了体现 IT 公司的特质，我们以 0 作为地面这一层的标记。
如果你某天在百度国际大厦的 0 层，仅可以选择 m 个电梯中的一个乘坐（不可以中途换电梯），请你计算，你按电梯中的按钮 n 次后（每次两个按钮选一个按），可以到达的最低楼层数。

输入

输入的第一行包括两个整数，分别为 n 和 m（1 ≤ n ≤ 1,000,000，1 ≤ m ≤ 2,000），表示按电梯按钮的次数和大厦中的电梯数量。接下去的 m 行，每行包

括 2 个由空格分割的数字，分别表示了提供的 m 个电梯中的某一个的上行按钮上升一次的层数 ui 和下行按钮下降一次的层数 di（1 ≤ ui,di ≤ 1000）

输出

输出一个正整数，表示选用 m 个电梯中的一个后，在电梯里按电梯中的按钮 n 次后（每次两个按钮选一个按），可以到达的最低楼层数。

样例输入

10 3
15 4
15 12
7 12

样例输出

13

提示

按钮上的移动楼层数无法改变，比方说从 8 层向下 9 层是不可行的

```
#include <cstdio>
#include <limits>
#include <algorithm>
#include <iostream>
using namespace std;

int core(int u, int d, int N)
{
  const long long m = (static_cast<long long>(N)*u-1LL)/(u+d);
  const long long n = N - m;
  int r = u*n - m*d;
  return r;
}

int main(){
   int n,m;
   int r=numeric_limits<int>::max();;
   cin>>n>>m;
   while(m--){
      int s,x;
      int temp;
      cin>>s>>x;
      temp=core(s,x,n);
      if(temp<r) r=temp;
   }
   cout<<r<<endl;
   return 0;
}
```