檔案處理

資料來源： http://www.tcgs.tc.edu.tw/~sagit/cpp/q4.htm

一、格式化輸出

在討論檔案的處理之前，我們先對輸出的格式化做個補充。先看到 C 的情況，只要在 %d 或 %f 的 % 後面加上格式化字元即可，下面是幾種常用的格式化表示法：

%4d — 印出寬度為 4 的整數

%.2f — 印出小數點後兩位的浮點數

%6.2f — 印出小數點後兩位，總寬度為 6 (含小數點)的浮點數

%03d — 印出三位數的整數，前面補 0

而 C++ 的話則是要先 #include <iomanip>，下面是幾個例子：

cout << setw(4) << a; — 印出寬度為 4 的數

cout << fixed << setprecision(2) << a;

— 印到小數點後兩位

cout << setfill('0') << setw(3) << a;

— 印出三位數的整數，前面補 0

前面提到的 setw、fixed 等我們稱之為 Manipulator，而要使用帶有參數的 Manipulator 則要 #include <iomanip>，下面我們列出幾個常用的 Manipulator ：

Manipulator	意義
flush	將輸出緩衝區的資料送出
endl	印出換行並送出 (flush)
setw(n)	設定列印寬度為 n
setprecision(n)	當 fixed 設定時，設定印到小數後第 n 位；而如果沒有設定 fixed，則是整個數的寬度(含整數部分以及小數點)
setfill(c)	以 c 為填充字元 (c 為字元 char)
fixed	以 xxx.xxx 的形式印出
scientific	以科學記號表示法印出
showpoint	印出小數後面的 0
uppercase	以大寫字母印出
left	靠左對齊
right	靠右對齊
dec	以十進位印出
hex	以十六進位印出

另外，因為 ACM 使用的 GCC 版本比較舊，不支援上面某些語法，因此如果題目要求輸出時要列印到小數點後三位，可以在程式一開頭使用下面語法：

```
cout.setf(ios::fixed);
cout.precision(3);
```

如果要以大寫字母印出，則是使用：

```
cout.setf(ios::uppercase);
```

至於 setw(n) 在 ACM 還是可以使用的，但是要記得 #include <iomanip> 。

二、檔案處理

之前我們看到的直接從鍵盤輸入資料，處理完之後把結果顯示在螢幕上，不過有些時候，當我們要輸入或輸出的資料很大的時候，我們會希望它輸入成一個檔案或是把輸出資料存成一個檔案，以方便我們使用。

不管是 C 還是 C++，要做檔案處理，都會經過下面的步驟：

1. 開啟檔案 (Open)，且檢查是否開啟成功
2. 對檔案做輸入、輸出
3. 關閉檔案 (Close)

我們先來看到 C 裡面開啟檔案和關閉檔案的寫法：

```
#include <stdio.h>
int main()
{
   FILE *fin, *fout;
   fin=fopen("pa.in", "rt");
   if(fin==NULL) ... //  開檔失敗
   fout=fopen("pa.out", "w+t");
   if(fout==NULL) ... //  開檔失敗
   ... //  輸入、輸出處理
   fclose(fin);
   fclose(fout);
   return 0;
}
```

我們要先宣告一種 FILE 的變數，而宣告的時候要以指標的型式宣告，也就是變數前面加一個星號 *，接下來用 fopen() 開啟檔案，第一個參數是要開啟的檔名，第二個參數是開啟的模式，第一個字母 r 代表讀取(Read)、w 代表寫入(Write)，還有一種是 a 代表附加(Append)，這種方式會把寫入的資料加到檔案後面。而第二個字母 t 代表以文字模式(Text)開啟，相對的還有一種是 b 則是以二進位模式開啟。

fopen() 執行完後，會傳回該檔案的指標，而如果開檔失敗，則是傳回 NULL，所以我們可以用 if(fin==NULL) 的方式檢查是否開檔失敗，而事實上 NULL 的值就是 0，所以也可以寫成 if(!fin)。

最後，當輸入、輸出處理完畢，我們要把開啟的檔案關閉，這時候使用的是 fclose() 函數，裡面的參數就是該檔案的指標。

至於怎麼從檔案讀取資料和寫入資料，我們是使用 fscanf()、fprintf() 函數，第一個參數先使用該檔案的指標，後面的部分和之前學過的 scanf()、printf() 幾乎一樣，例如：

```
fscanf(fin, "%d %d", &a, &b);
fprintf(fout, "%d\n", a+b);
```

此外，我們可以用 feof() 函數(End Of File)檢查輸入的檔案是否已經讀到檔案結束了，例如：

```
if(feof(fin)) ... //檔案結束
```

如果檔案還沒結束，則 feof() 函數傳回 0，否則傳回不是 0 的數。

下面我們把 C 常用的檔案處理函數做個整理：

fopen(檔案名稱, 模式) — 以指定模式開啟檔案

fclose(檔案指標) — 關閉檔案

fprintf(檔案指標, 格式字串, 參數一, 參數二, ...)
— 格式化輸出到檔案

fscanf(檔案指標, 格式字串, 變數一位址, 變數二位址, ...)
— 從檔案格式化輸入

feof(檔案指標) — 檢查是否已經讀到檔案結束

至於 C++ 的部分，則要先 #include <fstream>，開啟和關閉的程式如下：

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
   ifstream fin;
   ofstream fout;
   fin.open("pa.in");
   if( !fin.is_open() ) ... // 開啟失敗
   fout.open("pa.out");
   if( !fout.is_open() ) ... // 開啟失敗
   ... // 輸入、輸出處理
   fin.close();
   fout.close();
   return 0;
}
```

上面我們使用兩種物件， ifstream 用來輸入(Input)、ofstream 用來輸出(Output)，接下來用 open() 函數開啟檔案，也可以直接在宣告變數的時候指定檔名，例如：

```
ifstream fin("pa.in");
ofstream fout("pa.out");
```

接下來可以用 is_open() 函數檢查是否開啟成功，傳回非 0 的數則代表檔案開啟成功，失敗則是傳回 0。最後，我們可以用 close() 函數把檔案關閉。

至於檔案的輸入與輸出，則是和 cin、cout 類似，使用 <<、>> 等符號來操作，例如：

```
fin >> a >> b;
fout << a*b << endl;
```

至於要檢查是已經讀到檔案結束，則是用 eof() 函數，例如：

```
if( fin.eof() ) ... // 檔案結束
```

事實上，cin、cout 分別屬於 istream、ostream 物件，而 ifstream、ofstream 則又是從這兩種物件而來，所以很多操作是相通的。我們同樣把上面提到的函數做個整理：

open(檔案名稱) — 開啟檔案

close() — 關閉檔案

is_open() — 檢查檔案是否開啟

eof() — 檢查是否已經讀到檔案結束

三、將螢幕與鍵盤視為檔案

有時候我們會希望在寫程式的時候從鍵盤輸入、螢幕輸出，等到寫完程式後才改成以檔案處理。事實上，電腦本身即是把鍵盤及螢幕的輸入、輸出當成檔案來處理，在 DOS、Windows 系統下，只要把 CON (不分大小寫)當成檔名，即可把鍵盤當成輸入檔案、螢幕當成輸出檔案，例如：

```
ifstream fin("con");
fin << a;
```

上面的程式則會從鍵盤讀取資料到變數 a。等到程式寫完之後，再將 CON 改成真正的檔名即可。

四、freopen( ) 的使用

在某些程式設計比賽中(例如 NPSC)，所有的輸入及輸出都是到某個固定的檔案，不會有從鍵盤輸入、螢幕輸出以及多個檔案混合輸入輸出的情況，這種情況下，只要一開始時先針對鍵盤輸入、螢幕輸出的方式，以 C 來說就是使用 scanf( )、printf( )，以 C++ 來說就是使用 cin 、 cout ，等程式完成後，在 main( ) 的一開始加上下面這兩行即可：

```
freopen("pa.in", "rt", stdin);
freopen("pa.out", "w+t", stdout);
```

這兩行和之前用過的 C 開檔的語法是不是很像。其實 freopen( ) 的功能就是把原本已經開啟的檔案指標重新以另一個檔名開啟，而 stdin 、 stdout 是 C/C++ 程式一開始執行時自動開啟的， stdin 對應到標準輸入， 也就是鍵盤，而 stdout 對應到標準輸出，也就是螢幕。如此一來，程式也就不需要大幅修改，算是這類比賽時的一種小技巧。

五、練習題

1. Q10300：Ecological Premium
2. Q476：Points in Figures: Rectangles
3. Q477：Points in Figures: Rectangles and Circles

10300 - Ecological Premium

German farmers are given a premium depending on the conditions at their farmyard. Imagine the following simplified regulation: you know the size of each farmer's farmyard in square meters and the number of animals living at it. We won't make a difference between different animals, although this is far from reality. Moreover you have information about the degree the farmer uses environment-friendly equipment and practices, expressed in a single integer greater than zero. The amount of money a farmer receives can be calculated from these parameters as follows. First you need the space a single animal occupies at an average. This value (in square meters) is then multiplied by the parameter that stands for the farmer's environment-friendliness, resulting in the premium a farmer is paid per animal he owns. To compute the final premium of a farmer just multiply this premium per animal with the number of animals the farmer owns.

Input

The first line of input contains a single positive integer **n (<20)**, the number of test cases. Each test case starts with a line containing a single integer **f (0<f<20)**, the number of farmers in the test case. This line is followed by one line per farmer containing three positive integers each: the size of the farmyard in square meters, the number of animals he owns and the integer value that expresses the farmer's environment-friendliness. Input is terminated by end of file. No integer in the input is greater than **100000** or less than **0**.

Output

For each test case output one line containing a single integer that holds the summed burden for Germany's budget, which will always be a whole number. Do not output any blank lines.

Sample Input

```
3
5
1 1 1
2 2 2
3 3 3
2 3 4
8 9 2
3
9 1 8
6 12 1
8 1 1
3
10 30 40
9 8 5
100 1000 70
```

Sample Output

```
38
86
7445
```

MyNote:

476 - Points in Figures: Rectangles

Given a list of rectangles and a list of points in the *x-y* plane, determine for each point which figures (if any) contain the point.

Input

There will be $n(\leq 10)$ rectangles descriptions, one per line. The first character will designate the type of figure (``r'' for rectangle). This character will be followed by four real values designating the *x-y* coordinates of the upper left and lower right corners.

The end of the list will be signalled by a line containing an asterisk in column one.

The remaining lines will contain the *x-y* coordinates, one per line, of the points to be tested. The end of this list will be indicated by a point with coordinates 9999.9 9999.9; these values should not be included in the output.

Points coinciding with a figure border are not considered inside.

Output

For each point to be tested, write a message of the form:

Point i is contained in figure j

for each figure that contains that point. If the point is not contained in any figure, write a message of the form:

Point i is not contained in any figure

Points and figures should be numbered in the order in which they appear in the input.

Sample Input

r 8.5 17.0 25.5 -8.5

r 0.0 10.3 5.5 0.0

r 2.5 12.5 12.5 2.5

*

2.0 2.0

4.7 5.3

6.9 11.2

20.0 20.0

17.6 3.2

-5.2 -7.8

9999.9 9999.9

Sample Output

Point 1 is contained in figure 2

Point 2 is contained in figure 2

Point 2 is contained in figure 3

Point 3 is contained in figure 3

Point 4 is not contained in any figure

Point 5 is contained in figure 1

Point 6 is not contained in any figure

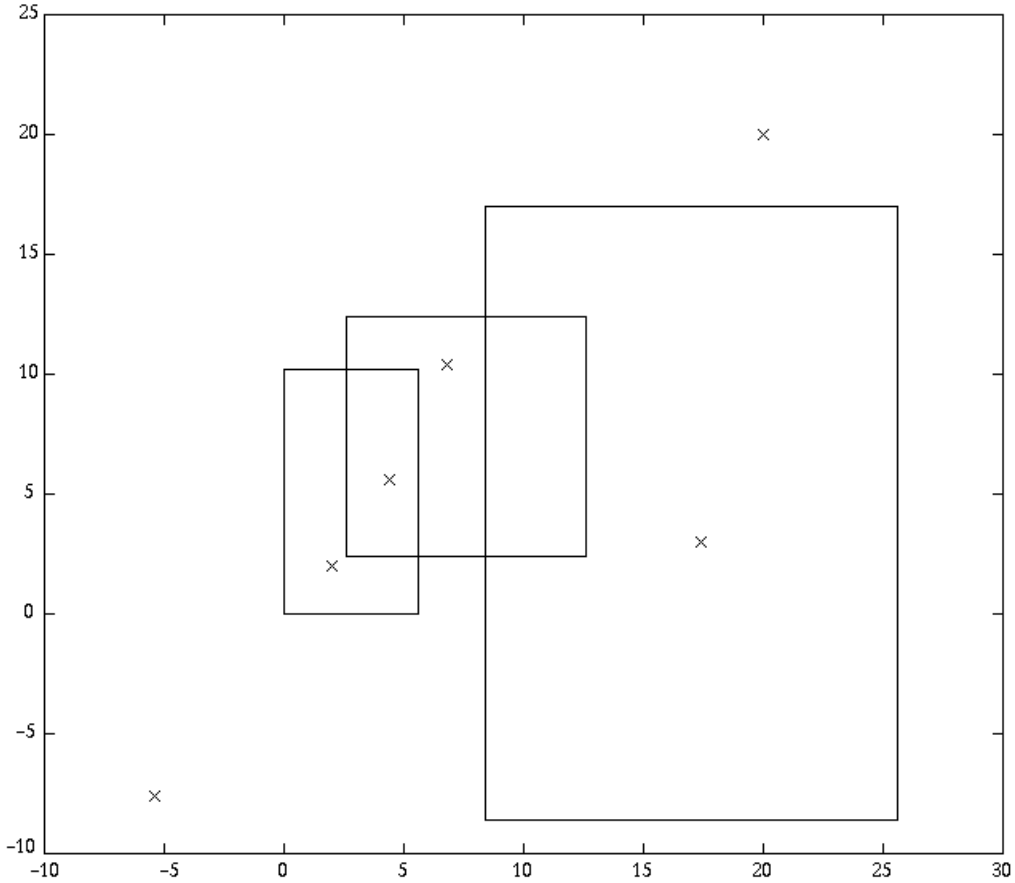

Diagrama of sample input figures and data points

MyNote: