**★☆☆☆☆**

**題組：Problem Set Archive with Online Judge**

**題號：10903：Rock-Paper-Scissors Tournament**

**解題者：黃粲富**

**解題日期：2013年3月21日**

**題意：**

n (1<= n <=100)位玩家舉辦猜拳遊戲，每位玩家會進行 k (1<= k <=100)場對決，總共有 k\*n\*(n-1)/2 場。計算每位玩家的勝率w/(w+x)，其中 w 為勝場數，x 為敗場數，而平手則不需計算。

 每行輸入 p1 m1 p2 m2 ，p 為玩家(1<=p1,p2<=n)，m 為出拳(Rock,Paper,Scissors)，0 為結束。

每行按玩家順序輸出勝率(計算至小數第三位)，若無勝負則輸出” – “，每個案例要隔行輸出。

**題意範例：**

input：

 2 3

⭠ 計算場次 3\*2\*(2-1)/2 = 3

 1 paper 2 rock

玩家1 勝:2 敗:1

玩家2 勝:1 敗:2

 2 paper 1 scissors

 2 scussors 1 paper

 3 1

⭠計算場次3\*1\*(3-1)/2 = 3

 1 rock 3 rock

玩家1 勝:1 敗:0 平:1

玩家2 勝:0 敗:1 平:1

玩家3 勝:0 敗:0 平:2

 2 paper 1 scissors

 2 paper 3 paper

 0

output：

玩家1 勝率=2/(2+1)=0.667

玩家2 勝率=1/(1+2)=0.333

 0.667

 0.333

⭠ 需間隔一行

 1.000

玩家1 勝率=1/(1+0)=1.000

玩家2 勝率=0/(0+1)=0.000

玩家3 勝率=0/(0+0) => -

 0.000

 -

⭠ 最後的case換行印完後，嚴禁再換行。

**解法：**用暴力法直接計算

**解法範例：**

 結構： (以上面input第二個case為例)

|  |  |  |  |
| --- | --- | --- | --- |
| 玩家 | P1 | P2 | P3 |
|  | 0 | 1 | 2 |
| Win | 0 | 0 | 0 |
| Lose | 0 | 0 | 0 |

用calloc分配並初始化陣列

 統計後

|  |  |  |  |
| --- | --- | --- | --- |
| 玩家 | P1 | P2 | P3 |
|  | 0 | 1 | 2 |
| Win | 1 | 0 | 0 |
| Lose | 0 | 1 | 0 |

注意事項：

 1. 印出時用%.3f

 2. case換行可用flag判斷

 3. 計算勝率時，應先判斷分母是否為零，在進行除法運算

**討論：**

有沒有其他更好的結構或統計方式？

心得：

 上傳時，得到好幾次WA，

 就是因為最後換行的格式錯誤。

**程式：**[註]因排版問題，並非完全相同，註解皆當程式碼用細明體下去排。

#include<stdio.h>

#include<stdlib.h>

struct player

{

 int win; //勝場數

 int lose; //敗場數

};

int judge(char\*,char\*); //判斷勝負之函式 參數為玩家1和玩家2所出拳

int main()

{

 int n,k,p1,p2; //n 玩家數; k 每位玩家出場數; p1 p2 為玩家

 //[註] p1 p2 指該場比賽之玩家

 //以下玩家1玩家2並非指第一或第二位玩家

 int j,i,a,w; //j 場數; i for參數; a 勝負場數和; w 敗場數

 int f=0; //f 分行flag 初始值為 0

 char s1[10],s2[10]; //玩家出拳(字串)

 struct player \*players; //玩家陣列指標名

 while(scanf("%d",&n)&&n!=0)

 {

 if(f==1)

 printf("\n");

 scanf("%d",&k);

 players = calloc(n,sizeof(struct player));

//calloc 分配記憶體並歸零

 j = k\*n\*(n-1)/2; //計算場次(接下來的輸入行數)

 for(;j>0;--j)

 {

 scanf("%d %s %d %s",&p1,s1,&p2,s2);

 switch(judge(s1,s2))

 {

 case 1 : //玩家1勝

 ++players[p1-1].win;

 ++players[p2-1].lose;

 break;

 case -1: //玩家2勝

 ++players[p2-1].win;

 ++players[p1-1].lose;

 break;

 }

 }

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

 {

 w = players[i].win; //玩家敗場數

 a = players[i].lose + w; //玩家總和場數(無平手)

 if(a==0)

 printf("-\n"); //若總和為零 直接輸出 -

 else

 printf("%.3f\n",((float)w)/a);

//否則 輸出結果至小數第三位

 }

 free(players); //釋放 calloc 分配之空間

 if(f==0)

 f = 1; //更改 flag 狀態

 }

 return 0;

}

int judge(char \*s1, char \*s2) // s1 玩家1出拳 ; s2 玩家2出拳

{

 if( (s1[0]=='s'&&s2[0]=='p') //[註] 因為字串是固定的所以直接
 ||(s1[0]=='p'&&s2[0]=='r') //判斷第一個字元，而未使用strcmp

 ||(s1[0]=='r'&&s2[0]=='s') )

 return 1; // 1 玩家1勝

 else if( (s2[0]=='s'&&s1[0]=='p')

 ||(s2[0]=='p'&&s1[0]=='r')

 ||(s2[0]=='r'&&s1[0]=='s') )

 return -1; // -1 玩家2勝

 else

 return 0; //平手

}