Programming Design, Spring 2013
Homework 12

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

To submit your work, please upload the following one file to the online grading system PDOGS at http://stella.im.ntu.edu.tw/online-judgement/.

1. A CPP file for Problem 1 (to the PD014 section).

NO hard copy and NO late submission. The due time of this homework is ***1:00pm, June 11, 2013***.

Problem 1

(100 points) Please write a C++ program according to the following instructions.

The bingo game

We are all familiar with the game "Bingo". While it has many different formats, in this homework we will focus on the following one (and its variations). First, a player put 1, 2, ..., and 25 into a five by five table, one number for each cell. She can arrange these numbers in any order she likes. Then another player (called a "coordinator") starts to call several numbers randomly (the amount of numbers to be called is determined by the coordinator). Once a number is called, the player circle or box that number. Her score is the number of vertical, horizontal, or diagonal lines that contain five circled or boxed numbers. We say such a line is "complete". For example, if Nancy constructs Table 1 and and the coordinator calls numbers 1, 2, 3, ..., and 15, then Nancy's table will become Table 2. As there are two complete lines (the fourth row and the northwest-southeast diagonal), Nancy's score is 2. When multiple players play this game together, each of them will construct her own table and the one earning the highest score is the winner.

1	19	24	7	9
2	8	6	20	23
5	12	3	25	22
15	11	4	14	10
18	16	17	21	13

Table 1: Nancy's table at the beginning

$\boxed{1}$	19	24	$\boxed{7}$	$\boxed{9}$
$\boxed{2}$	$\boxed{8}$	$\boxed{6}$	20	23
$\boxed{5}$	$\boxed{12}$	$\boxed{3}$	25	22
$\boxed{15}$	$\boxed{11}$	$\boxed{4}$	$\boxed{14}$	$\boxed{10}$
18	16	17	21	$\boxed{13}$

Table 2: Nancy's table at the end

In this homework, you need to implement the above basic Bingo game and two variations. In the first variation, a player gets one extra point for each three-by-three square in each all nine numbers are crossed out. In the example above, Nancy's score will become 3 because there is one three-by-three square satisfying this condition. In the second variation, each vertical or horizontal complete line counts for 1 point, but a complete diagonal line counts for 2 points. In Nancy's case, her score will become 3 because the complete diagonal now gives her 2 points. For the above three ways of counting the scores, we call them "the basic game", "square extension", and "diagonal extension", respectively.

Overview of your task

Your program is going to simulate a group of people playing this game. The number of people in the group is unknown but cannot be greater than 100. In the first n lines of the input data, you will be given information for n players playing the basic game. Then in ten lines you will be notified that ten coordinators come in and each of them calls several numbers for the n players to play. These numbers called by coordinators will be given to you and you need to decide who are the winners for the ten basic games. After m more lines of information given to you for players playing the matrix extension, another fifteen coordinators come in and initiate fifteen rounds. You need to decide the fifteen winners among the $n+m$ players. Finally, p more players join the competition by playing the diagonal extension and then ten coordinators initiate the last ten rounds. You need to find the ten winners among the $n+m+p$ winners.[1] In total there are 35 rounds and you need to make 35 times of output to display the 35 winners.[2]

Input and output formats

There are four types of input:

- A player's information: In a line of a player, first there is a character `B`, `M`, or `D`, then a white space, then a string with no space as the player's name, then a white space, and finally a single integer as a random seed s. The way of constructing the player's table, which is identical for all three game formats, is as follows. First, the table is initialized to be $A_{ij} = 5(i-1)+j$ for all $i, j = 1, ..., 5$, where A_{ij} is the number at row i and column j. After the random seed s is used as an argument for the C++ function `srand()`, `rand()` is invoked for 50 times to generate 50 random integers r'_1, r'_2, ..., and r'_{50}. By applying the formula

  $$r_i = r'_i \mod 5 + 1,$$

  we will have $r_i \in \{1, 2, ..., 5\}$ for all $i = 1, ..., 50$. For a given i, if i is odd, then row r_i is "shifted" to the right by one cell, i.e., $A_{i,2}$ becomes $A_{i,1}$, $A_{i,3}$ becomes $A_{i,2}$, ..., and $A_{i,1}$ becomes $A_{i,5}$. If i is even, then column r_i is "shifted" downwards by one cell, i.e., $A_{2,j}$ becomes $A_{1,i}$, $A_{3,i}$ becomes $A_{2,i}$, ..., and $A_{1,i}$ becomes $A_{5,i}$. As an example, suppose a line contains

  ```
  P Cindy 100
  ```

  for a player called Cindy, then 100 is used as a random seed and the sequence of random numbers are $r'_1 = 365$, $r'_2 = 1216$, $r'_3 = 5415$, $r'_4 = 16704$, ..., and $r'_{50} = 13079$. These random numbers then result in $r_1 = 1$, $r_2 = 2$, $r_3 = 1$, $r_4 = 5$, ..., and $r_{50} = 5$. Cindy's table can thus be generated by running the above shifting for 50 times, where the first four times of shifting are listed below:

  $$\begin{matrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{matrix} \rightarrow \begin{matrix} 5 & 1 & 2 & 3 & 4 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{matrix} \rightarrow \begin{matrix} 5 & 22 & 2 & 3 & 4 \\ 6 & 1 & 8 & 9 & 10 \\ 11 & 7 & 13 & 14 & 15 \\ 16 & 12 & 18 & 19 & 20 \\ 21 & 17 & 23 & 24 & 25 \end{matrix}$$

  $$\rightarrow \begin{matrix} 4 & 5 & 22 & 2 & 3 \\ 6 & 1 & 8 & 9 & 10 \\ 11 & 7 & 13 & 14 & 15 \\ 16 & 12 & 18 & 19 & 20 \\ 21 & 17 & 23 & 24 & 25 \end{matrix} \rightarrow \begin{matrix} 4 & 5 & 22 & 2 & 25 \\ 6 & 1 & 8 & 9 & 3 \\ 11 & 7 & 13 & 14 & 10 \\ 16 & 12 & 18 & 19 & 15 \\ 21 & 17 & 23 & 24 & 20 \end{matrix} \rightarrow \cdots$$

  Note that given the same random seed, your program should always generate the same table. In the testing data, no two players will be given the same random seed. With the table, the player will count her scores with the rule for the basic game, matrix extension, or diagonal extension if the first character is `B`, `M`, or `D`, respectively.

- A coordinator's numbers for a game: In this case, a line will contain a character `G`, then a white space, then several nonrepeating numbers between 1 and 25, where two consecutive numbers are separated by a white space.

[1]As we mentioned, you do not know the exact value of n, m, and p. However, you know $n+m+p \leq 100$.

[2]Please note that the three different groups of players all have five-by-five tables. However, when they are called to play, they need to use different rules to count their scores.

For lines starting with `B`, `M`, or `D`, no output should be displayed. For a line starting with `G`, the winner's name and her scores should be displayed, with a white space separating them. When there is a tie, the one who earns the highest scores and enters the game first is the winner.

Example

Suppose you are given the following testing data

```
B Linda 5
B May 10
B Rose 20
G 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
M Ruby 200
G 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 25
D Lily 100
G 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 24
```

The five tables for Linda, May, Rose, Ruby, and Lily are

$$\begin{bmatrix} 21 & 25 & 2 & 15 & 24 \\ 22 & 11 & 5 & 4 & 7 \\ 18 & 12 & 14 & 13 & 23 \\ 6 & 20 & 9 & 3 & 19 \\ 10 & 17 & 1 & 8 & 16 \end{bmatrix}, \begin{bmatrix} 11 & 25 & 5 & 20 & 7 \\ 4 & 13 & 10 & 17 & 14 \\ 9 & 19 & 15 & 12 & 18 \\ 22 & 16 & 24 & 6 & 21 \\ 8 & 3 & 2 & 1 & 23 \end{bmatrix}, \begin{bmatrix} 18 & 13 & 8 & 23 & 2 \\ 6 & 4 & 15 & 20 & 10 \\ 19 & 11 & 1 & 5 & 16 \\ 17 & 3 & 14 & 24 & 12 \\ 7 & 21 & 9 & 25 & 22 \end{bmatrix},$$

$$\begin{bmatrix} 15 & 12 & 4 & 7 & 2 \\ 5 & 24 & 8 & 6 & 21 \\ 14 & 25 & 3 & 17 & 11 \\ 13 & 18 & 9 & 22 & 23 \\ 16 & 1 & 20 & 19 & 10 \end{bmatrix}, \text{and} \begin{bmatrix} 24 & 16 & 4 & 25 & 19 \\ 10 & 3 & 18 & 23 & 14 \\ 13 & 21 & 7 & 9 & 12 \\ 22 & 20 & 17 & 11 & 15 \\ 1 & 5 & 6 & 8 & 2 \end{bmatrix},$$

respectively. Your output should be

```
Linda 2
Linda 2
Lily 3
```

Please note that whenever a coordinator initiates a game, Linda always counts her scores based on the rule for the basic game, Ruby always counts her scores based on the rule for the matrix extension, etc.

Some suggestions

Write three classes, one for each game. Use inheritance to simply the work for writing the three classes Use polymorphism so that you may store these games into a single array. For each class, write a function with the same name that calculates the scores of a given set of called numbers. Set the function we just mentioned to be a virtual function (why?).

Grading criteria

Your program will be graded based on the following criteria:

- 70% of your grades for this program will be based on the correctness of your output. The online grading system will input a set of testing data, which includes many lines of events. Among all the events, 35 lines will require outputs. You may only see the grades of running your program on these data but cannot see the inputs and outputs. The 35 output lines count for 70 points, i.e., 2 points for each line.
- 30% of your grades for this program will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.