Assignment 4: Dots and Boxes

Due: 20:00, Wed 12 Oct 2016 Full marks: 100

Introduction

In this assignment, you will implement a two-player paper-and-pencil game called *Dots and Boxes*. Starting with an empty grid of dots, two players take turns adding one horizontal or vertical line between two un-joined adjacent dots. A player who completes the fourth side of a 1×1 box earns one point *and* takes an extra turn. The game ends when the grid is full, and the player with more points wins. We shall assume that the game is played in a cruciform grid, so that there are five boxes when the grid is full. (Therefore, there will never be a draw game.) Figure 1 shows an example grid. We use the symbol o to denote dots, and the symbols - - and | to denote the lines of boxes. In the figure, the right-sided box is already formed (assumed to be by Player 1), and the left-sided box is one vertical line from being formed.

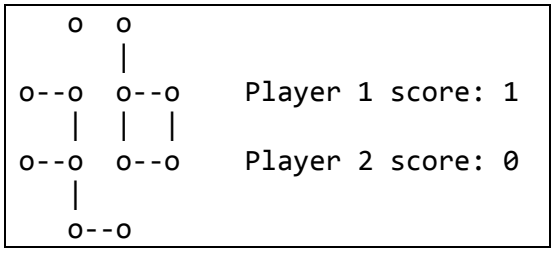

Figure 1: An Example Dots and Boxes Configuration

Program Specification

This section describes the representation of a grid in a game, the necessary C++ functions, and the flow of your program.

Grid Representation

There are 16 possible positions that we can place a line in the cruciform grid. Therefore, we use integers 1 to 16 to denote these positions, as illustrated in Figure 2. The positions are basically ordered top-to-bottom, left-to-right.

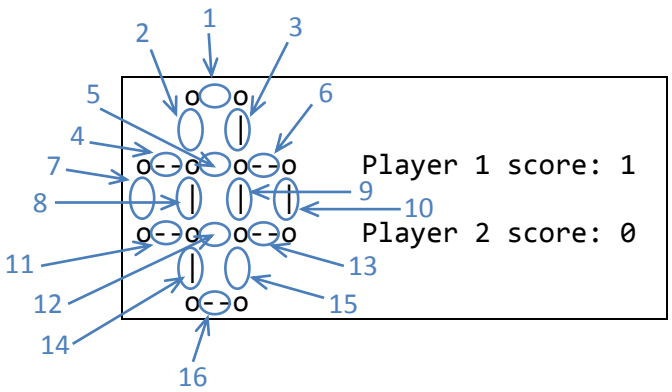

Figure 2: Numbers for Line Positions

To encode the whole grid and the players' scores in a game, we use an 18-digit integer $d_1d_2d_3d_4d_5d_6d_7d_8d_9d_{10}d_{11}d_{12}d_{13}d_{14}d_{15}d_{16}d_{17}d_{18}$. The first 16 digits $d_1 \ldots d_{16}$ are either 0 or 1, denoting whether the corresponding positions 1 to 16 are empty or filled. The last two digits d_{17} and d_{18} are between 0 and 5 (inclusive) to denote the scores of Players 1 and 2 respectively. For example, the grid in Figure 1 can be encoded by the integer 001101011110110110, which also stores the scores of Player 1 (1) and Player 2 (0). Using this representation, an empty grid (with no lines filled) is simply encoded as the integer 000000000000000000. (Note: in C++, integer constants should *NOT contain leading zeroes*, otherwise, they will be treated as octal numbers. Therefore, an empty grid is specified as 0 only in C++.) A full grid (with lines filled in all positions) is encoded as 1111111111111111*xy*, where *x* and *y* depend on the scores of Players 1 and 2. For example, 111111111111111123 is a game won by Player 2.

The data type `int` in C++ is not big enough to store an 18-digit integer. In your program, you have to use a bigger integer type called `long long`. In Visual Studio, `long long` is a 64-bit signed integer type, whose range is -9,223,372,036,854,775,808 … 9,223,372,036,854,775,807.

Provided and Required Functions

Your program must contain the following six functions. You can design extra functions for your program if you find necessary. Note that you are *NOT allowed to declare any global variables (variables declared outside any functions)*.

`void displayGrid(long long grid)`

This function prints `grid` to the screen using the format in Figure 1.

`bool isFilled(long long grid, int pos)`

Returns `true` if position `pos` of `grid` is filled with a line; returns `false` otherwise.

`int playerScore(long long grid, int p)`

Returns the score of Player p in `grid`. (Either the 17th or 18th digits in `grid`.)

`void updateGrid(long long &grid, int pos, int p)`

Performs the task of Player p putting a line in position `pos` in `grid`. The grid should get updated, and if any new boxes are formed, the score of Player p should be incremented. Note that putting one line can form at most two boxes. The *reference parameter* `grid` should store the updated grid when the function returns. To determine if a new box is formed, calling the `isFilled(…)` function is useful.

`int gameState(long long grid)`

Returns an integer 1, 2, or 3, denoting the game state of `grid` listed below:

Game State	Meaning
1	Game is finished and Player 1 wins.
2	Game is finished and Player 2 wins.
3	Game is not yet finished. (That is, grid is not full.)

`int main()`

The entry point of your program execution.

In the above functions, you can assume that (a) the parameter `grid` is always a proper encoding of a grid of the game; (b) the parameter `pos` is always between 1 and 16 (inclusive); and (c) the parameter `p` is either 1 or 2.

Program Flow

The following shows the flow of your program. You should call the functions in the above to aid your implementation.

1. The program starts the game with an empty grid, and Player 1 takes the first turn.
2. Then, you should prompt the current player to enter a position to put a line in. You can assume that *the player always enters an integer*.
3. In case the player makes an invalid input (outside the range 1 to 16 or the input position was already filled), display a warning message and go back to step 2.
4. Update the grid by putting a line in the position.
5. If the current player has formed new box(es), print a message and keep him/her the current player. Otherwise, swap the other player to become the current player.
6. Repeat steps 2 to 5 until the grid is full. (That is, until game is over.)
7. Once the grid is full, determine the winner or a draw and display the message "Player 1 wins!" or "Player 2 wins!" accordingly. (Recall that there can be no draw games.)

Program Output

The following shows some sample output of the program. The **bold blue** text is user input and the other text is the program output. You can try the provided sample program for other input. *Your program output should be exactly the same as the sample program* (i.e., same text, same symbols, same letter case, same number of spaces, etc.). Otherwise, it will be considered as *wrong*, even if you have computed the correct result.

```
   o  o

o  o  o  o    Player 1 score: 0

o  o  o  o    Player 2 score: 0

   o  o
Player 1, make your move (1-16): 10↵
   o  o

o  o  o  o    Player 1 score: 0
         |
o  o  o  o    Player 2 score: 0

   o  o
Player 2, make your move (1-16): 4↵
   o  o

o--o  o  o    Player 1 score: 0
         |
o  o  o  o    Player 2 score: 0
```

```
   o  o
Player 1, make your move (1-16): 20↵
Invalid move! Try again.
Player 1, make your move (1-16): -5↵
Invalid move! Try again.
Player 1, make your move (1-16): 4↵
Invalid move! Try again.
Player 1, make your move (1-16): 15↵
   o  o

o--o  o  o    Player 1 score: 0
         |
o  o  o  o    Player 2 score: 0
      |
   o  o
Player 2, make your move (1-16): 5↵
   o  o

o--o--o  o    Player 1 score: 0
         |
o  o  o  o    Player 2 score: 0
      |
   o  o
Player 1, make your move (1-16): 2↵
   o  o
   |
o--o--o  o    Player 1 score: 0
         |
o  o  o  o    Player 2 score: 0
      |
   o  o
Player 2, make your move (1-16): 13↵
   o  o
   |
o--o--o  o    Player 1 score: 0
         |
o  o  o--o    Player 2 score: 0
      |
   o  o
Player 1, make your move (1-16): 7↵
   o  o
   |
o--o--o  o    Player 1 score: 0
|        |
o  o  o--o    Player 2 score: 0
      |
   o  o
Player 2, make your move (1-16): 3↵
   o  o
   |  |
o--o--o  o    Player 1 score: 0
|        |
```

```
o  o  o--o    Player 2 score: 0
      |
   o  o
Player 1, make your move (1-16): 1↵
Player 1 scores! Gets an extra turn.
   o--o
   |  |
o--o--o  o    Player 1 score: 1
|        |
o  o  o--o    Player 2 score: 0
      |
   o  o
Player 1, make your move (1-16): 12↵
   o--o
   |  |
o--o--o  o    Player 1 score: 1
|        |
o  o--o--o    Player 2 score: 0
      |
   o  o
Player 2, make your move (1-16): 8↵
   o--o
   |  |
o--o--o  o    Player 1 score: 1
|  |     |
o  o--o--o    Player 2 score: 0
      |
   o  o
Player 1, make your move (1-16): 14↵
   o--o
   |  |
o--o--o  o    Player 1 score: 1
|  |     |
o  o--o--o    Player 2 score: 0
   |  |
   o  o
Player 2, make your move (1-16): 9↵
Player 2 scores! Gets an extra turn.
   o--o
   |  |
o--o--o  o    Player 1 score: 1
|  |  |  |
o  o--o--o    Player 2 score: 1
   |  |
   o  o
Player 2, make your move (1-16): 16↵
Player 2 scores! Gets an extra turn.
   o--o
   |  |
o--o--o  o    Player 1 score: 1
|  |  |  |
o  o--o--o    Player 2 score: 2
   |  |
```

```
   o--o
Player 2, make your move (1-16): 11↵
Player 2 scores! Gets an extra turn.
   o--o
   |  |
o--o--o  o    Player 1 score: 1
|  |  |  |
o--o--o--o    Player 2 score: 3
   |  |
   o--o
Player 2, make your move (1-16): 6↵
Player 2 scores! Gets an extra turn.
   o--o
   |  |
o--o--o--o    Player 1 score: 1
|  |  |  |
o--o--o--o    Player 2 score: 4
   |  |
   o--o
Player 2 wins!
```

Submission and Marking

- Your program file name should be `dotsboxes.cpp`. Submit the file in Blackboard (https://elearn.cuhk.edu.hk/).
- Insert your name, student ID, and e-mail address as comments at the beginning of your source file.
- You can submit your assignment multiple times. Only the latest submission counts.
- Your program should be *free of compilation errors and warnings*.
- Your program should *include suitable comments as documentation*.
- *Plagiarism is strictly monitored and heavily punished if proven.* Lending your work to others is subjected to the same penalty as the copier.