Assignment 5: Pah Tum

Due: 23:59, Tue 22 Nov 2022 File name: `pahtum.cpp` Full marks: 100

Introduction

The objective of this assignment is to practice the use of 2-D arrays with loops. You will write a program to play a two-player board game called *Pah Tum* (古西亞連棋).

This game, originated from Mesopotamia (now Iraq) at least 3800 years ago, is one of the oldest board games in history. It is played on a 7×7 square board, but can be generalized to any board size. At the beginning of the game, some squares called *black holes* are closed off for play, and all other squares are empty. Two players alternate turns to place their pieces on the empty squares, trying to create as many lines as possible horizontally ↔ or vertically ↕. A line must be at least three squares long to score points. The longer a line, the more points are scored. The exact scoring system is as follows:

Pieces in a Line	3	4	5	6	7
Points	3	10	25	56	119

(When played on a large board, longer lines can be formed to score even higher points.)

When the board is full, the player with more points is the winner. Note that the number of playable squares after putting black holes should be even, so that each player can place the same number of pieces eventually. Figure 1 shows an example game board of Pah Tum. We use the symbols '**.**' and '#' to denote an empty square and a black hole respectively. The two players are 'O' and 'X'. In the game, player O has (a) a vertical line of three pieces in column C (3 points); (b) a vertical line of four pieces in column E (10 points); and (c) a horizontal line of three pieces in row 3 (3 points). Thus player O has $3 + 10 + 3 = 16$ points. Player X has two lines of three pieces in column B and row 4, and thus has a score of $3 + 3 = 6$ points.

```
   A B C D E F G
 0 . O X # O . .
 1 . X O X O # .
 2 . X O X O . .
 3 . X O O O X .
 4 . O X X X O .
 5 . . X . # # .
 6 . . . . # . .
O score: 16
X score: 6
```

Figure 1: An Example 7×7 Pah Tum Game Board

Program Specification

This section describes some required named constants, the game board representation, program flow, and some special requirements.

Named Constants

Your program shall define the following global named constants.

```
// Global
const int N = 7;    // Board size
const int SCORES[16] = {0, 0, 0, 3, 10, 25, 56, 119, 246, 501, 1012, 2035,
                        4082, 8177, 16368, 32751};    // Scoring table
const int BLACKHOLES[16] = {0, 0, 0, 0, 0, 0, 0, 5, 6, 9, 10, 13, 14, 17,
                            20, 23};    // Num of black holes
```

- The named constant N is the board size, that is, the number of rows and columns of the board. Your program shall be *scalable* and still *work normally for other board size*. When grading, we may modify your program by changing N to any values in the range 7–15 for testing.
- The constant array SCORES is the scoring table storing the points scored when a line of certain length is formed. E.g., `SCORES[6]` is 56, which is the points scored for a line of length 6.
- The constant array BLACKHOLES stores the number of black holes to be put to a board of a certain size. E.g., `BLACKHOLES[13]` is 17, which is the number of black holes needed for a 13×13 board.

Game Board Representation

The game board shall be represented by a two-dimensional N × N array of `char`.

```
// Local
char board[N][N];
```

The array contents shall be either 'O', 'X', '#' (black hole), or '.' (empty). The elements `board[0][0]`, `board[0][N-1]`, `board[N-1][0]`, and `board[N-1][N-1]` denote the top-left, top-right, bottom-left, and bottom-right corners of the board respectively.

Program Flow

1. The program starts with an empty board and asks the user to enter *some black hole positions*. The number of black holes needed is `BLACKHOLES[N]`. (N is an array index.) You can assume that *each position is always a letter followed by an integer*, denoting the column and row position of the board. Both uppercase and lowercase letters are allowed. E.g., both `C 5` and `c 5` denote the same position.
2. When the user enters an invalid position (out of board range) or the input position is already a black hole previously entered, the program shall display a warning message. This phase finishes when enough different black hole positions are entered.
3. After marking all the black holes, player O moves first.
4. Then, prompt the current player to make a move. Again, *the input is always a letter followed by an integer*, denoting the column and row position of the board, with uppercase and lowercase letters both allowed. E.g., `F 6` and `f 6`.
5. In case the player enters an invalid position (out of board range) or the input position is not empty, display a warning message and go back to Step 4.
6. Update the board by putting the player's mark (O or X) in the input position. Also update the score (if any) of the player.
7. Repeat steps 4–6 until the board is full. Alternate players O and X in each round.
8. Once the game is finished, display the messages *"Player O wins!"*, *"Player X wins!"*, or *"Draw game!"* accordingly.

Special Requirements

- *Global variables (variables declared outside any functions) are not allowed*. Nonetheless, `const` ones (e.g., N, SCORES, BLACKHOLES) do not count.
- Your program shall be decomposed into *at least four functions* (including `main()`). At least *two functions* shall have *array parameter(s)*.
- Your program should be *scalable* to other values for the named constants N. That is, your program shall still *work normally for other board size*. When grading, we may modify your program by changing N to any values in the range 7–15 for testing.

Tips

- In the printing of the board in Figure 1, the row numbers are printed with a width of 2 (`setw(2)`). That is, if a row number is single-digit, then there shall be a space before the row number. If a row number is double-digit, then there shall be no space before the row number.
- Placing one piece to a square may form two scoring lines, one horizontal and one vertical, forming a "T-junction" or an "L-junction".
- A previously formed line may be lengthened. E.g., if a player previously formed a line of four squares (10 points), s/he can lengthen this line by placing a piece to the left or right end of the line (if possible) to make it a line of five squares (25 points). In this case, the score of the player is increased by 15 points only (10 → 25).
- Placing one piece to a square may join two previously formed lines to become one longer line. E.g., given a row …OO.OOO…, placing an O in the empty space in the middle joins the OO and OOO lines ($0 + 3 = 3$ points) to become one …OOOOOO… line (56 points). In this case, the score of Player O is increased by 53 points (3 → 56).

Sample Run

In the following sample run, the blue text is user input and the other text is the program printout. *More sample runs* are provided in *Blackboard*. Besides, you can try the provided sample program for other input. *Your program output should be exactly the same as the sample program* (same text, symbols, letter case, spacings, etc.).

```
   A B C D E F G
 0 . . . . . . .
 1 . . . . . . .
 2 . . . . . . .
 3 . . . . . . .
 4 . . . . . . .
 5 . . . . . . .
 6 . . . . . . .
O score: 0
X score: 0
Enter 5 black hole positions:
C 3↵
E 2↵
C 1↵
H 3↵
```

```
Invalid. Try again!
C 3↵
Invalid. Try again!
g 5↵
G 0↵
   A B C D E F G
 0 . . . . . . #
 1 . . # . . . .
 2 . . . . # . .
 3 . . # . . . .
 4 . . . . . . .
 5 . . . . . . #
 6 . . . . . . .
O score: 0
X score: 0
Player O, make your move: G 5↵
Invalid. Try again!
Player O, make your move: e 1↵
   A B C D E F G
 0 . . . . . . #
 1 . . # . O . .
 2 . . . . # . .
 3 . . # . . . .
 4 . . . . . . .
 5 . . . . . . #
 6 . . . . . . .
O score: 0
X score: 0
Player X, make your move: E -1↵
Invalid. Try again!
Player X, make your move: E 3↵
   A B C D E F G
 0 . . . . . . #
 1 . . # . O . .
 2 . . . . # . .
 3 . . # . X . .
 4 . . . . . . .
 5 . . . . . . #
 6 . . . . . . .
O score: 0
X score: 0
```

⋮ *(Some moves are skipped to save space. See Blackboard for full version.)*

```
   A B C D E F G
 0 O . . . . . #
 1 X . # O O . O
 2 X O . O # . O
 3 O X # . X X .
 4 . O . X . . .
 5 . . . . X . #
 6 O O X . X . X
O score: 0
X score: 0
Player X, make your move: D 3↵
   A B C D E F G
 0 O . . . . . #
 1 X . # O O . O
 2 X O . O # . O
 3 O X # X X X .
 4 . O . X . . .
 5 . . . . X . #
 6 O O X . X . X
O score: 0
X score: 3
       ⋮      (Some moves are skipped to save space. See Blackboard for full version.)
   A B C D E F G
 0 O X X . . . #
 1 X . # O O X O
 2 X O O O # . O
 3 O X # X X X .
 4 . O . X . . X
 5 O O . O X . #
 6 O O X X X O X
O score: 6
X score: 6
Player O, make your move: A 4↵
   A B C D E F G
 0 O X X . . . #
 1 X . # O O X O
 2 X O O O # . O
 3 O X # X X X .
 4 O O . X . . X
 5 O O . O X . #
 6 O O X X X O X
O score: 16
X score: 6
       ⋮      (Some moves are skipped to save space. See Blackboard for full version.)
```

```
   A B C D E F G
 0 O X X X O X #
 1 X O # O O X O
 2 X O O O # . O
 3 O X # X X X X
 4 O O O X O X X
 5 O O O O X X #
 6 O O X X X O X
O score: 29
X score: 19
Player X, make your move: F 2↵
   A B C D E F G
 0 O X X X O X #
 1 X O # O O X O
 2 X O O O # X O
 3 O X # X X X X
 4 O O O X O X X
 5 O O O O X X #
 6 O O X X X O X
O score: 29
X score: 72
Player X wins!
```

Submission and Marking

- Your program file name should be `pahtum.cpp`. Submit the file in Blackboard (https://blackboard.cuhk.edu.hk/).
- Insert your name, student ID, and e-mail as comments at the beginning of your source file.
- Besides the above information, your program should further *include suitable comments as documentation*.
- You can submit your assignment multiple times. Only the latest submission counts.
- Your program should be *free of compilation errors and warnings*.
- ***Do NOT plagiarize.*** Sending your work to others is subjected to the same penalty as the copying student.