CSCI1540 Fundamental Computing with C++, Fall 2019/20
Department of Computer Science and Engineering, The Chinese University of Hong Kong

Assignment 5: Game of Hex

Due: 20:00, Thu 21 Nov 2019 File name: `hex.cpp` Full marks: 100

Introduction

The objective of this assignment is to practice the use of arrays with loops. You will write a program using arrays to play a game called *Hex* (六貫棋), which is a two-player game played on an initially empty diamond game board, with a typical size 11×11. The two players have some chess pieces in two different colors. They take turns to place their pieces on empty cells of the board. The goal of the first player is to form a chain of his/her pieces connecting the left and right border of the board, while the goal of the second player is to form a chain of his/her pieces connecting the top and bottom border. The four corners belong to both of the two adjacent borders. The player who first achieves his/her goal wins. This game always has a winner; it never ends in a draw.

We shall call the first and second players 'O' and 'X' respectively. Figure 1 shows an example game board. The character '.' denotes an empty cell. The rows and columns are named in numbers (0–10) and letters (A–K) respectively. Each cell has at most ***six neighbors***. For example, cell F3 has neighbors F2, G2, E3, G3, E4, and F4. (E2 and G4 are ***not*** neighbors of F3.)

```
 A B C D E F G H I J K
0 O X . X . . . . . . .
 1 . . . . . X . . . . .
  2 X . O . . . . . O X .
   3 . . . . . O . . . O X
    4 . X . . . . . . . . .
     5 O X X . O . . X . . O
      6 O . . . . . . . . X O
       7 . . . . . . . . . . .
        8 . O . . . . . X . . X
         9 . X . O . . . O . . .
         10 O . X . . . . . . O .
```

Figure 1: An Example Game Board

Program Specification

Game Board

You can use a two-dimensional array of `char` to represent the game board.

```
const int N = 11;
……
char board[N][N];
```

The array elements `board[0][0]`, `board[0][10]`, `board[10][0]`, and `board[10][10]` denote four corner cells A0, K0, A10, and K10 respectively. Note that among the eight neighbors of an array element (not on the borders), only six of them are true neighbors of the corresponding cell location. For example, E2 (`board[2][4]`) and G4 (`board[4][6]`) are ***not*** neighbors of F3 (`board[3][5]`).

Game Flow

1. Starting with an empty board, players O and X make moves alternately. Player O makes the first move.
2. In each move, prompt the player to enter two inputs denoting the cell location to be placed. You can assume that *the inputs are always a character followed by an integer*.
3. A user input is *invalid* if: (a) it is not a proper cell location (i.e., rows 0–10 and columns A–K, *big letters only*), *or* (b) the input location is already occupied.
4. You should warn the player about invalid inputs and prompt the same player to enter again until a valid input is entered.
5. When a player has formed a winning chain of pieces (left-right for player O and top-bottom for player X), print the message "Player O wins!" or "Player X wins!" accordingly.

Hint for Checking Winning Chain

After a player has made a move, you have to check whether s/he has made a winning chain of pieces. To do so, you may need *another two-dimensional array* of type `bool` to store whether each cell location is "reachable" from the input location. If there is a cell on one side of the board reachable from the input location, and another cell on the opposite side also reachable from the input location, then the player has formed a winning chain. Figure 2 illustrates the idea for player O. After a player placed a piece on the board, the whole reachability array is `false` except the input position, which is `true`. Then, you incrementally expand the reachability (set `true`) to neighbors of the input position, to neighbors of neighbors of the input position, to neighbors of neighbors of neighbors of the input position, and so on. A player has formed a winning chain if you identify that the two corresponding borders are reached. Your challenge is to write code to correctly compute the two-dimensional reachability array every time a player has placed a piece in the board.

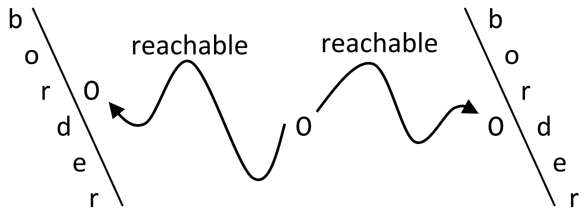

Figure 2: A Winning Chain of Pieces for Player O

Special Requirements

- *Global variables (variables declared outside any functions) are not allowed*. Nonetheless, `const` ones (e.g., N) do not count.
- Your program should be decomposed into *at least four functions* (including `main()`). *At least two functions* should have *array parameter(s)*.

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.). Note that *there is a space after the ':' in the program printout*.

```
  A B C D E F G H I J K
 0 . . . . . . . . . . .
  1 . . . . . . . . . . .
   2 . . . . . . . . . . .
    3 . . . . . . . . . . .
     4 . . . . . . . . . . .
      5 . . . . . . . . . . .
       6 . . . . . . . . . . .
        7 . . . . . . . . . . .
         8 . . . . . . . . . . .
          9 . . . . . . . . . . .
          10 . . . . . . . . . . .
Player O moves: A 0↵
  A B C D E F G H I J K
 0 O . . . . . . . . . .
  1 . . . . . . . . . . .
   2 . . . . . . . . . . .
    3 . . . . . . . . . . .
     4 . . . . . . . . . . .
      5 . . . . . . . . . . .
       6 . . . . . . . . . . .
        7 . . . . . . . . . . .
         8 . . . . . . . . . . .
          9 . . . . . . . . . . .
          10 . . . . . . . . . . .
Player X moves: B 11↵
Invalid! Try again.
Player X moves: M 4↵
Invalid! Try again.
Player X moves: A 0↵
Invalid! Try again.
Player X moves: b 9↵
Invalid! Try again.
Player X moves: B 9↵
  A B C D E F G H I J K
 0 O . . . . . . . . . .
  1 . . . . . . . . . . .
   2 . . . . . . . . . . .
    3 . . . . . . . . . . .
     4 . . . . . . . . . . .
      5 . . . . . . . . . . .
       6 . . . . . . . . . . .
        7 . . . . . . . . . . .
         8 . . . . . . . . . . .
          9 . X . . . . . . . . .
          10 . . . . . . . . . . .
    ⋮    (Many moves skipped to save space. Full version available in Blackboard.)
  A B C D E F G H I J K
 0 O X O O X O O X . X X
  1 O O X X O X O X X X O
   2 X O X O X X O X X O X
    3 O O O X O O O . O O O
     4 X O O O . X X X O X X
      5 X . X O O O O X . O X
       6 O O X O O X O O . O O
        7 . X . X X O O O X O X
         8 X X X O X X X O X O O
          9 X X X X X O O O X X X
          10 O X X O X X O O X O O
Player X moves: I 0↵
  A B C D E F G H I J K
 0 O X O O X O O X X X X
  1 O O X X O X O X X X O
   2 X O X O X X O X X O X
    3 O O O X O O O . O O O
     4 X O O O . X X X O X X
      5 X . X O O O O X . O X
       6 O O X O O X O O . O O
        7 . X . X X O O O X O X
         8 X X X O X X X O X O O
          9 X X X X X O O O X X X
          10 O X X O X X O O X O O
Player O moves: I 5↵
  A B C D E F G H I J K
 0 O X O O X O O X X X X
  1 O O X X O X O X X X O
   2 X O X O X X O X X O X
    3 O O O X O O O . O O O
     4 X O O O . X X X O X X
      5 X . X O O O O X O O X
       6 O O X O O X O O . O O
        7 . X . X X O O O X O X
         8 X X X O X X X O X O O
          9 X X X X X O O O X X X
          10 O X X O X X O O X O O
Player O wins!
```

Submission and Marking

- Your program file name should be hex.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.