Assignment 4: Tant Fant

Due: 20:00, Thu 7 Nov 2019 File name: `tantfant.cpp` Full marks: 100

Introduction

The objective of this assignment is to practice (1) defining functions (being a callee) and (2) calling functions (being a caller). You will implement a game called *Tant Fant*. Two players, 1 and 2, play the game on a 3×3 board. The top and bottom rows of the board are called the *home rows* of players 1 and 2 respectively. Each player has three pieces, which are initially placed at their own home rows. The players take turn to move one of their pieces to an *adjacent* empty position. The player who first aligns his/her three pieces *horizontally*, *vertically*, or *diagonally*, except his/her home row, wins. Figure 1 shows a game board and some example game plays. The numbers 1 and 2 denote the players, and `0` denotes an empty space. Two positions are adjacent if they are connected by a line. At the end of Figure 1, player 2 forms a horizontal line at the middle row and wins the game.

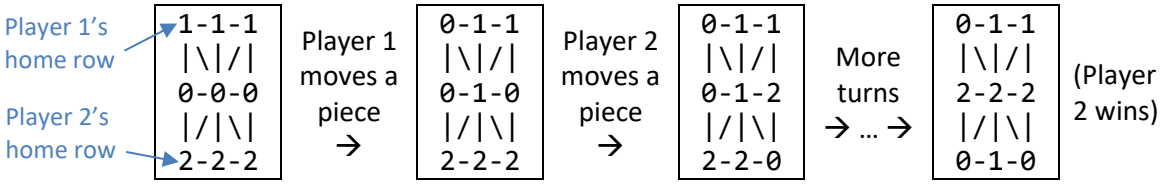

Figure 1: An Example Tant Fant Game Play

Program Specification

This section describes the requirements, program design, function decomposition, game flow, etc.

Basic Requirements

You *cannot declare any global variables (variables declared outside any functions)*. You *cannot use any functions in the* `<cmath>` library. You *cannot use any arrays* in your code.

Board Representation

There are nine possible positions in a board. Therefore, we use integers 1–9 to denote these board positions, as illustrated in Figure 2. The positions are basically ordered top-to-bottom, left-to-right.

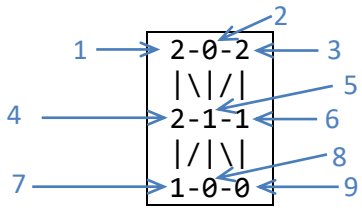

Figure 2: Position Numbers for Board Positions

To encode the whole game board, we use a 9-digit integer $d_1d_2d_3d_4d_5d_6d_7d_8d_9$. Each digit d_i is either 0, 1, or 2. Values 1 or 2 mean position i is a piece of players 1 or 2 respectively. Value 0 means position i is empty. Using this representation, the initial board (with both players in their own home rows) is encoded as 111000222. As an example, the last board in Figure 1 is encoded as 11222010. (Note: in C++, integer constants should *NOT contain leading zeroes*, otherwise, they will be treated as octal numbers. Therefore, the last board is not 011222010 but 11222010.) The data type `int` in C++

is typically 32-bit and thus big enough to store a 9-digit integer. Therefore, we can simply use one `int` variable to store the board configuration of a game.

Provided and Required Functions

Your program must contain the following functions. *Some of them are written for you already (Provided) and you should not modify their contents.* The others will be written by you (Required). These functions must be called *somewhere* in your program. You must *not* modify the prototypes of all these functions. You can design extra functions if you find necessary.

In the functions below, you can assume that (a) parameter `board` is always a proper encoding of a game board (9-digit integer; each shall be 0–2); (b) parameters `pos`, `from`, and `to` are always 1–9; and (c) parameter p is always either 1 or 2. (That is, you do not have to check these conditions in these functions. Any of these checks should be done elsewhere, e.g., in program flow described later.)

(Provided) `int status(int board, int pos);`

Returns the *status* of position `pos` of the game `board`. That is, returns 0 when the position is empty; returns 1 when the position is player 1's piece; and return 2 when the position is player 2's piece.

(Provided) `void printBoard(int board);`

Prints the `board` to the screen using the format in Figure 1. When writing this function, calling the `status()` function would be helpful.

(Required) `void updateBoard(int &board, int from, int to, int p);`

This function updates the `from`th digit of the *reference parameter* `board` to 0, *and* the `to`th digit of `board` to digit p. The other digits of `board` remain unchanged. The following shows some sample function calls and the expected results.

board	from	to	p	Value of board *after* calling updateBoard(board, from, to, p)
202211100	6	2	1	212210100
100211220	4	9	2	100011222
2011122	8	4	2	2211102

This function is useful in the task of player p moving a piece from position `from` to position `to` of `board` in the game. At the end of this function, the value of `board` becomes the new board configuration *after* the piece movement. Note that you do *not* have to check in this function whether the piece movement is valid or not. (You check validity elsewhere. See Step 3 of the Program Flow section later.) You just have to update the digits in the two relevant positions.

(Required) `int formLine(int board)`

This function checks if any player has formed a line horizontally, vertically, or diagonally in `board` *except* his/her home row. If so, the function returns either 1 or 2, meaning the player of the line. Otherwise, the function returns 0, meaning no line is formed. For example, the call `formLine(101010222)` should return 0; `formLine(102010221)` should return 1; and `formLine(102012012)` should return 2. When writing this function, calling the `status()` function would be helpful. You can assume that no two players can form such lines simultaneously.

Program Flow

The program flow for the game is described as follows. You should call all the functions above to aid your implementation. All user inputs mentioned below are assumed to be always integers.

1. The game starts with a board where both players are in their own home rows. Player 1 takes the first turn.
2. Prompt the current player to enter *two integers* to specify that s/he wants to move a piece *from a source position to a destination position*.
3. A user input is invalid if (a) two input positions are not 1–9, (b) the source position is not occupied by the current player, (c) the destination position is not empty, *or* (d) the source and destination positions are not adjacent. In case it is invalid, display a warning message and go back to Step 2.
4. Update the board by moving the source piece to the destination.
5. Repeat steps 2–4 with alternate players 1 and 2, until a player has formed a line vertically, horizontally, or diagonally, except in his/her home row.
6. Once the game is finished, display the message *"Player 1 wins!"* or *"Player 2 wins!"* accordingly.

Sample Run

In the following sample run, the blue text is user input and the other text is the program printout. 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*.

```
1-1-1
|\|/|
0-0-0
|/|\|
2-2-2
Player 1 (from to): 3 5↵
1-1-0
|\|/|
0-1-0
|/|\|
2-2-2
Player 2 (from to): 0 4↵
Invalid. Try again!
Player 2 (from to): 2 3↵
Invalid. Try again!
Player 2 (from to): 6 3↵
Invalid. Try again!
Player 2 (from to): 7 5↵
Invalid. Try again!
Player 2 (from to): 8 6↵
Invalid. Try again!
Player 2 (from to): 9 6↵
1-1-0
|\|/|
0-1-2
|/|\|
2-2-0
Player 1 (from to): 1 4↵
```

Source position is outside 1–9

Position 2 is not a piece of player 2

Position 6 is empty; not a piece of player 2

Position 5 has been occupied by player 1

Source and destination are not adjacent

```
0-1-0
|\|/|
1-1-2
|/|\|
2-2-0
Player 2 (from to): 6 3↵
0-1-2
|\|/|
1-1-0
|/|\|
2-2-0
Player 1 (from to): 2 1↵
1-0-2
|\|/|
1-1-0
|/|\|
2-2-0
Player 2 (from to): 8 9↵
1-0-2
|\|/|
1-1-0
|/|\|
2-0-2
Player 1 (from to): 5 8↵
1-0-2
|\|/|
1-0-0
|/|\|
2-1-2
Player 2 (from to): 9 5↵
1-0-2
|\|/|
1-2-0
|/|\|
2-1-0
Player 2 wins!
```

Submission and Marking

- Your program file name should be `tantfant.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.
- 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*.
- ***Do NOT plagiarize.*** Sending your work to others is subjected to the same penalty as the copying student.