ENGG1110 Problem Solving by Programming (2022-2023 Term 2) Project

Deadline: 21-Apr-2023 (Fri) 23:59

---

1. Introduction

Notakto is a game played by two players on multiple 3x3 game boards. It is said to be invented by Bob Koca [1].

In this project, the following game rules are used.

1. There are only two 3x3 game boards, which are empty initially.
2. Two players take turns to choose a game board and place a cross (X) on an unoccupied cell.
3. A game board is dead if it has three-in-a-row of crosses (in horizontal, vertical or diagonal).
4. Players cannot choose a dead game board to place a cross.
5. Players cannot skip their turns.
6. When there is only one non-dead game board left, the player who places the last cross to make it dead (i.e., having three-in-a-row of crosses) loses the game. In other words, the other player is the winner.

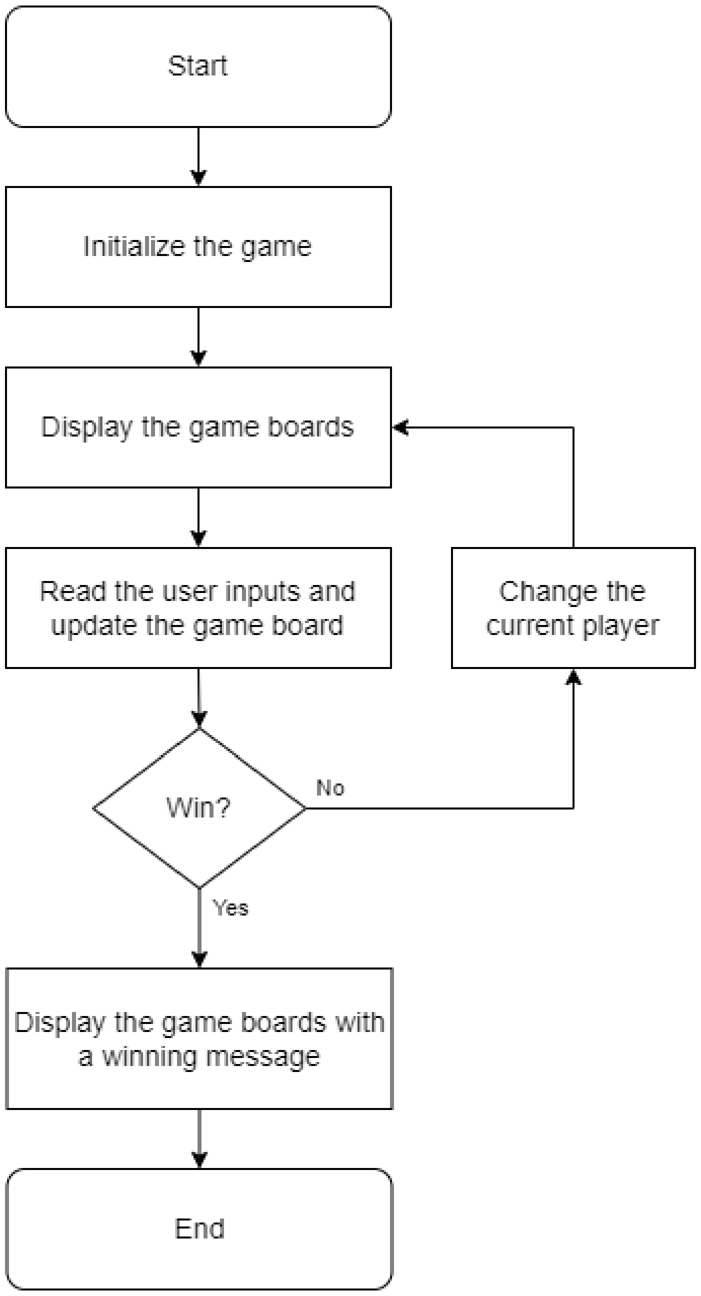
Your task is to implement the aforesaid game for one or two human players in C language. If there is only one human player, he/she will play with a computer player.

You are required to complete the given **main.c** without modifying any existing code (except uncommenting the given source code for testing) nor any function signature. Marks will be deducted for each modification.

[1] Wikipedia contributors, "Notakto," Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Notakto&oldid=1037737456 (accessed March 7, 2023).

2. Program Flow

The following diagram shows the program flow for two human players:


Start
Initialize the game
Display the game boards
Read the user inputs and update the game board
Change the current player
Win?
No
Yes
Display the game boards with a winning message
End

3. Schedule

The suggested schedule is shown as follows:

Week	Tasks
11	• Complete the following functions: ○ **initGameBoard()** ○ **printGameBoard()** ○ **isGameBoardDead()** • No submission is needed at this stage
12-13	• Complete the following functions: ○ **placeCrossByHumanPlayer()** ○ **updateGameBoard()** • Test the completed functions so far by uncommenting the given source code in **main()** • Complete the **main()** function for the game played by two human players • Test the program by using the sample runs • Test the program by using your own test cases • No submission is needed at this stage
14-15	• Complete the following functions: ○ **placeCrossByComputerPlayer()** ○ **countNumOfCrosses()** • Complete the **main()** function for the game played by one human player • Test the program by using the sample runs • Test the program by using your own test cases • Submit your code by the deadline

4. Detailed Program Design

4.1 Header Files and Variable Declarations

No header files other than stdio.h are allowed. In addition, no global variables (those declared outside functions) are allowed. In other words, all variables must be declared inside functions. Marks will be deducted for each violation.

4.2 Initializing Game Boards

The **main()** function invokes the **initGameBoard()** function twice to initialize the two game boards one by one. Please design your own representation for the game board. For example:

- Does **gameBoard[0][0]** represent the top left corner or the bottom left corner of the game board?
- Which numbers represent empty cells and crosses?

4.3 Reading Number of Human Players

The **main()** function prints the following message to ask for the number of human players:

```
Enter the number of human players [1-2]:
```

Here you can assume that the user must be valid (either 1 or 2).

- If there are two human players, they are Player 1 and Player 2.
- If there is only one human player, he/she is Player 1 and another player is the computer player.

For both cases, Player 1 moves first.

4.4 Printing Game Boards

In the beginning of each turn, the **main()** function invokes the **printTwoGameBoards()** function, which displays the two game boards on the screen. Empty cells are displayed with numbers (similar to a NumPad, having 1 at the bottom left corner and 9 at the top right corner). Occupied cells are displayed with crosses. The following shows some examples in a game.

In the beginning of a game, all cells in the two game boards are empty.	`# Game Board 1 #`
	`|7|8|9|`
	`|4|5|6|`
	`|1|2|3|`
	`# Game Board 2 #`
	`|7|8|9|`
	`|4|5|6|`
	`|1|2|3|`

After 4 moves in the game (Game Board 2 is dead)	`# Game Board 1 #`
	`|7|8|9|`
	`|4|X|6|`
	`|1|2|3|`
	`# Game Board 2 #`
	`|X|8|9|`
	`|4|X|6|`
	`|1|2|X|`

After 9 moves in the game (Both game boards are dead and the game ends)	`# Game Board 1 #`
	`|X|8|9|`
	`|X|X|X|`
	`|X|X|3|`
	`# Game Board 2 #`
	`|X|8|9|`
	`|4|X|6|`
	`|1|2|X|`

In the project, you are <u>required</u> to follow exactly the above output format. Using other output format will result in mark deduction.

The **printTwoGameBoards()** function has been implemented for you already and no change is needed. Instead, please complete the **printGameBoard()** function that will be called by the **printTwoGameBoards()** function.

4.5 Checking Whether the Game Board is Dead

The **isGameBoardDead()** function returns 1 if the specified game board is dead ((i.e., having three-in-a-row of crosses). Otherwise, it returns 0.

This function will be called by various functions later and it should <u>not</u> print any message (except debugging messages for your own use, but please delete them before submission).

4.6 Placing Cross by Human Player

Depending on who the current player is, the **main()** function prints one of the following messages:

```
# Player 1's turn #
```

```
# Player 2's turn #
```

and then invokes the **placeCrossByHumanPlayer()** function, which first prints the following message to ask the current human player to choose the game board:

```
Choose the game board:
```

You can assume that the user inputs must be integers. But the user input may not be valid. When the input is not 1 or 2, print the following message and read the input again:

```
Input out of range. Please input again:
```

When the chosen game board is dead, print the following message and read the input again:

```
The chosen game board is dead. Please input again:
```

The above checking will be repeated until the input is valid.

After a valid game board is chosen, the **placeCrossByHumanPlayer()** function will invoke the **updateGameBoard()** function, which first prints the following message to ask the current human player to place a cross on the chosen game board.

```
Choose the cell:
```

You can assume that the user inputs must be integers. But the user input may not be valid. When the input is not in the range of 1 to 9, print the following message and read the input again:

```
Input out of range. Please input again:
```

When the chosen cell is occupied (i.e., has a cross already), print the following message and read the input again:

```
The chosen cell is occupied. Please input again:
```

The above checking will be repeated until the input is valid. Note that the game board will not be chosen again if the input for the cell is invalid.

After a valid cell is chosen, the **updateGameBoard()** function will update the chosen game board accordingly.

4.7 Placing Cross by Computer Player

When it is the computer's turn, the **main()** function prints the following message:

```
# Computer's turn #
```

and then invokes the **placeCrossByComputerPlayer()** function, which determines the next move of the computer player and updates the corresponding game board accordingly. The choices of the game board and the cell will also be printed on the screen with the following messages:

```
Choose the game board:
```

```
Choose the cell:
```

In this project, you are <u>required</u> to implement the following strategy (<u>using other strategies will result in mark deduction</u>):

- Choosing the game board
  - <u>Case A</u>: Both game boards are <u>not</u> dead (i.e., <u>not</u> having three-in-a-row of crosses)
    - If the first game board has fewer crosses than the second game board, choose the first game board.
    - Otherwise, choose the second game board.
  - <u>Case B</u>: One of the game boards is dead
    - Choose the other one that is not dead.
- Placing the cross
  - <u>Case A</u>: Both game boards are <u>not</u> dead
    - If the <u>**first**</u> game board was chosen:
      - Place the cross on the unoccupied cell with the <u>**smallest**</u> number (no matter whether this move will make the game board dead or not). For example: (Computer's move: Cell 1 on Game Board 1)

```
# Game Board 1 #
|7|8|9|
|4|5|6|
|1|2|3|
# Game Board 2 #
|7|8|9|
|4|5|6|
|X|2|3|
# Computer's turn #
Choose the game board:
1
Choose the cell:
1
```

    - If the <u>**second**</u> game board was chosen:
      - Place the cross on the unoccupied cell with the <u>**largest**</u> number (no matter whether this move will make the game board dead or not). For example: (Computer's move: Cell 7 on Game Board 2)

```
# Game Board 1 #
|X|8|9|
|4|X|6|
|X|2|3|
# Game Board 2 #
|7|X|X|
|4|5|6|
|1|2|3|
# Computer's turn #
Choose the game board:
2
Choose the cell:
7
```

- Case B: One of the game boards is dead
  - If the **first** game board was chosen:
    - Place the cross on the unoccupied cell with the **smallest** number that will not make the game board dead (i.e., not to lose the game). For example: (Computer's move: Cell 4 on Game Board 1)

```
# Game Board 1 #
|7|8|9|
|4|5|6|
|X|2|X|
# Game Board 2 #
|X|X|X|
|4|5|6|
|1|2|3|
# Computer's turn #
Choose the game board:
1
Choose the cell:
4
```

    - If all possible moves will make the game board dead, place the cross on the unoccupied cell with the **smallest** number (i.e., lose the game). For example: (Computer's move: Cell 3 on Game Board 1)

```
# Game Board 1 #
|7|X|X|
|X|5|X|
|X|X|3|
# Game Board 2 #
|7|8|9|
|4|5|6|
|X|X|X|
# Computer's turn #
Choose the game board:
1
Choose the cell:
3
```

- If the **<u>second</u>** game board was chosen:
    - Place the cross on the unoccupied cell with the **<u>largest</u>** number that will <u>not</u> make the game board dead (i.e. <u>not</u> to lose the game). For example: (Computer's move: Cell 5 on Game Board 2)

```
# Game Board 1 #
|X|8|9|
|X|5|6|
|X|X|3|
# Game Board 2 #
|7|X|X|
|4|5|6|
|1|2|X|
# Computer's turn #
Choose the game board:
2
Choose the cell:
5
```

    - If all possible moves will make the game board dead, place the cross on the unoccupied cell with the **<u>largest</u>** number (i.e., lose the game). For example: (Computer's move: Cell 7 on Game Board 2)

```
# Game Board 1 #
|7|8|9|
|X|5|6|
|X|X|X|
# Game Board 2 #
|7|X|X|
|X|X|6|
|1|2|X|
# Computer's turn #
Choose the game board:
2
Choose the cell:
7
```

4.8 Checking Whether the Game Ends

At the end of each turn, the **main()** function checks whether the game ends. If yes, display the two game boards, print one of the following messages according who the winner is:

```
Congratulations! Player 1 wins!
```

```
Congratulations! Player 2 wins!
```

```
Computer wins!
```

and terminate the program. If no, the game continues to the next player's turn.

5. Sample Run

Some sample runs are provided on Blackboard. In addition, you are strongly recommended to test your program by designing your own test cases.

6. Academic Honesty and Declaration Statement

Attention is drawn to University policy and regulations on honesty in academic work, and to the disciplinary guidelines and procedures applicable to breaches of such policy and regulations. Details may be found at https://www.cuhk.edu.hk/policy/academichonesty/.

You are required to place the following declaration statement as the comment in the beginning of the .c source code and fill in your information.

```
/**
 * ENGG1110 Problem Solving by Programming
 *
 * Course Project
 *
 * I declare that the project here submitted is original
 * except for source material explicitly acknowledged,
 * and that the same or closely related material has not been
 * previously submitted for another course.
 * I also acknowledge that I am aware of University policy and
 * regulations on honesty in academic work, and of the disciplinary
 * guidelines and procedures applicable to breaches of such
 * policy and regulations, as contained in the website.
 *
 * University Guideline on Academic Honesty:
 *   https://www.cuhk.edu.hk/policy/academichonesty/
 *
 * Student Name  : <your name>
 * Student ID    : <your student ID>
 * Class/Section : <your class/section>
 * Date          : <date>
 */
```

7. Grading Platform

We will grade your work in **Code::Blocks** under **Windows 10**.

8. Submission

Please follow the steps below to submit your work by the deadline.

1. Login Blackboard
2. Go to 2022R2 Problem Solving By Programming (ENGG1110X), where X is your class/section
3. Go to Project → Project Submission
    - Rename the modified **main.c** to **project_<your student ID>.c**
        - E.g., project_1155123456.c
    - Submit the .c file

Note: Please do not submit other files used in your IDE (if any).

Resubmissions are allowed. But only the latest one will be graded. 30% of the marks will be deducted for late submissions within three days. Late submissions more than three days will not be graded.