Principles of computer programming

Final project

Project task:

Write a program that implements a computer version of the connect four (https://en.wikipedia.org/wiki/Connect_Four) game. The game is played on a board that is 7 fields wide and 6 fields high. It is played by two players with the symbols ‘X’ and ‘O’. Players can choose the column in which their symbol is placed on top of the symbols that are already in that column. The winner is the player that first makes a row of 4 symbols either vertically, horizontally or diagonally. The goal is to write a program that implements this game. When started the program should display main menu with the following three options:

1.) Play a new game
2.) Load already saved game
3.) Exit the game

If option **1.) Play a new game** is chosen, the program should allow players to enter their names. First the ’X’ player should make its move by entering an integer number from 1 to 7 which indicates the row in which he would like to place his symbol. After this, the program should switch the player turns and allow each player to enter its sign to the board. After each turn, program should print out the state of the board. Game ends either if one of the players connects 4 symbols in a column or if the whole board is filled with signs and no one won. Program should check these ending conditions after each turn and properly inform the players if the game has ended and who won. If someone wins, program should print out the board with the higlighted connected symbols that caused the victory. This could be done by printing the board with ’Y’ symbols on the places of the connected symbols. After the game has ended, players should be asked if they would like to start a new game or go back to the main menu. If during the play a player selects a row that is already filled with symbols, program should inform him that such a move is not allowed and ask him to select some other row. If at any point one of the players enters 0 instead of an integer 1 to 7, program should save the current game. Game is saved by saving the current state of the board and the player names. Each new saved game should have a unique ID of the saved game. After saving the game, program should inform the players that their gaim has been saved and print the ID of the saved game, after this the game play should continue in the described way without further interuption.

If option **2.) Load already saved game** is chosen, a new menu opens with the following options:

1.) List all saved games
2.) List all saved games for a particular player
3.) Show the board of one of the saved games

4.) Load a game
5.) Return to main menu

If **1.) List all saved games** is selected, program should list all the games that are saved in the file. Games should be lsited is the following format:

ID, 'X' player name, 'O' player name, number of free fields left on the board

Each saved game should be printed in a new line

If **2.) List all saved games for a particular player** is selected, program should ask the user to enter a name and then it should list all the saved games that have that name as one of the players in the same format as specified above.

If **3.) Show the board of one of the saved games** is selected, program should ask the user to enter a an ID of a saved game and then print out the player names and the board for that game.

If **4.) Load a game** is selected, program should ask the user to enter an ID of the saved game and then start the game from the saved point. From this point on, the program should behave in the same way as when a new game is played. Once a loaded game has finished, program should allow the players to either start a new game (with empty board) or to return to main menu – i.e. the same choices that they have when they play a game from an empty board.

If in options 3.) and 4.) user selects a non-existing ID, program should inform him about this and ask him to enter a valid ID.

Delivery:

Students should mail their solution in a form of text file containing executable code, or a .zip file containing full CodeBlocks project. Code should contain comments that explain the logic of the solution. Solution should be mailed to the professor by e-mail: mtanaskovic@singidunum.ac.rs latest by Thursday 25.01.2018 at 12:00 (noon). Solutions that are not sent by this deadline will not be considered.

Defense:

In order to get the points for the final project, students have to defend their solution. Defence consists of practical demonstration that their code works, short oral explanation of what they did and a minor modification of the code on the spot.

Defence will be organized on Friday 26.01.2018. Students will be informed about the exact time of their defense in a response mail that they will get after submitting their solution.

Grades:

Students will get the points for their solution based on its accuracy and the knowledge that they show during defense. Maximal number of points that can be achieved is 30.

For any further questions regarding this project, you may contact the professor by e-mail: mtanaskovic@singidunum.ac.rs