Final Project
Connect Four

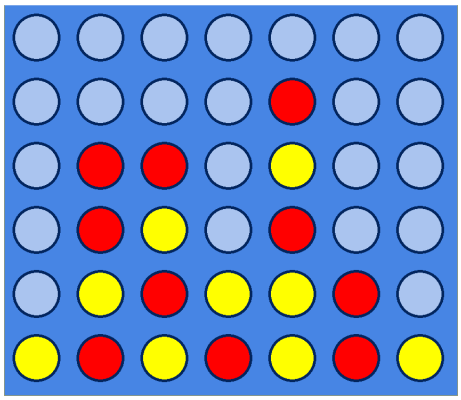

Figure 1: example on connect four of 7 x 6 board

1 Game Description

Connect 4 is a two-player game in which the players first chooses a color and then takes turns dropping their colored discs from the top into a grid. The pieces fall straight down, occupying the next available space within the column. The objective of the game is to connect-four of one's own discs of the same color next to each other vertically, horizontally, or diagonally. The two players keep playing until the board if full. The winner is the player having greater number of connected-fours. You can read on wikipedia.

2 Overview

The goal of the game is to connect connect-four discs of the same color next to each other vertically, horizontally, or diagonally as you can. The board may be of any size. Each player is assigned a random different color.

Score is calculated based on the number of connected fours. The game ends when the board is full. The winner is the player with the highest score.

The map is a two dimensional grid where it has no discs at all. Each player can select a column to drop a disc and the players should connect four discs to win.

3 Game Mode

Your game should have two game modes:

- Two player mode: Human vs. Human
- One player mode: Human vs. Computer

Any computer player strategy is acceptable even a random computer player that chooses any valid random column for the move

4 Grid Display

- You should display an empty grid with only borders between the rows and columns at the beginning of the game.
- Implement the game in text mode using coloured ASCII characters, e.g "X" for player1 and "O" for player2.
- It is up to you to implement your own display, but keep in mind that it should be easy to understand and playable!

5 User Interface

- During the game play you should print extra information beside the grid.
  - Whose player turn is.
  - Number of moves for each player.
  - Scores of each players.
  - Time passed in minutes and seconds.
- All the above fields should be updated each time any player makes a move.
- You are required to color the above information with the color of each player to make it more readable.
- You should provide some sort of an in-game menu in order to handle undo, redo, save, and exit.
- Take the input from user with any other convenient method to handle the selected column and the menu option.

6 End Game

The game ends when the board is full, **NOT** when any connected four is formed. The player with the highest score is the winner. When the game ends, keep the board, ask the player for his name and display his score and rank. Then, ask the player to either go back to the menu or exit.

7 User's Rank

- When a player successfully wins a game. You should take his name and display his score and rank. You shouldn't take name of the other player.
- Player's name should be case insensitive.
- High scores should be stored sorted in a file, and loaded when the game is started.
- If the player records a score higher than any of the high scores, the new high score is recorded in the high scores file then all high scores are displayed sorted.
- If it's the first time for the player, then add the user to the scores list (a file). Else, update the score if the current score is higher than the saved score.
- Number of accesses to the file to record a new high score and replace an old score should be minimized.
- The game can print the highest X scores, where X is a parameter to the game, given as input when the game starts.

8 Game Parameters

- Game parameters:
  - Board's height
  - Board's width
  - Maximum number of high scores (size of the high scores list)
- The configurations file should be in XML format. Here is a sample configuration file:

```
<Configurations>
    <Height>7</Height>
    <Width>9</Width>
    <Highscores>10</Highscores>
</Configurations>
```

- All white spaces in the configuration file should be ignored.
- For incorrect file format, display a message indicating so, and ask the user for correct file path to read it. If loading or parsing the file failed for 3 times, then load the default values: Width = 7, Height = 9, Highscores = 10

9 Undo and Redo

- The player can undo the game till the first move or redo till the last move.
- After undo, if the player make a new move, all the redo should be cleared.

10 Save and Load

- At any time, the game can be saved to a file.
- A saved game could be loaded and continued.
- Players names and scores should be saved to a file.
- No need to save undo data.
- You should save up to 3 games.
- Player should be able to select which game to load

Hint: You should minimize size of the file by saving a binary file, instead of a text file.

11 Main Menu

The main menu of your program should contain the following options :

- Start a new game
  - choose game mode vs. human or vs. computer
- Load a game
  - choose the saved game to load
- Top players
- Quit

Feel free to add other options.

12 Game Flow

1. Read game parameters from the configuration file. If the file doesn't exist or corrupted, then ask the user for the file path to read it. If loading the file failed for few times, then load the default values.
2. Show the main menu. For the game mode, ask the user for one or two players and then assign random colors for each player.
3. Loop till the end of the game:
   (a) Read a move from Player1
   (b) Check if it is valid input and if the selected column has an empty slot
   (c) If not valid, ask for another move
   (d) Do the move
   (e) Update the UI and player1's score
   (f) Repeat the same logic for player2. If it is "human vs. computer", then instead of reading the move as input, you should execute some logic for the computer's turn.
4. Update list of high scores if any and display the sorted list of high scores at the end of the running game.
5. Ask the player to either go back to the menu or exit.

13 Implementation Notes

- You are required to implement the game using the C programming language.
- **Top Priority** Your program **MUST NOT** crash under any circumstances, even against malicious users!
- All users' inputs should be validated. Any missed validation will reduce your marks.
- Take care of the following:
  - Naming conventions.
  - Code organization and style.
  - Code comments.
  - Split your code into meaningful functions.
- You have to use structure player, which will hold the player name, color, score, ...etc and structure configuration which will hold height, width and highscores.

14 Bonus

- Split your project on different files and use header files (different files for: ranking, game loop, computer turn, user interface, undo, ...).
- Use git to co-work with your teammate.
- Graphical User Interface (GUI).
- Implementing a convenient computer turn algorithms (AI, ...).
- Adding as an option in the game menu to print the best move using an AI algorithm (only in case of Human vs. Human mode).
- You are welcome to add extra features to your game.

15 Deliverables

- Demo:
  - You must provide a 3 minute (maximum) "voice over video" demo showing the implemented requirements, any additional bonus features, and short sample runs.
- Source code and executable:
  - Complete project and source files
  - Executable version of your project (the *.exe file)
- Report containing the following:
  - Description: You should briefly describe the application you implemented.
  - Features: Write down the features of the application.
  - Design Overview: Make a small overview on your design and the overall structure with minimal details.
  - Assumptions: Tell us if you had taken any assumption during the design or the implementation phase. necessary to be clarified.
  - Data Structure: Write down the data structure you have used in the program, like arrays, structures, ...
  - Description of the important functions/modules: Describe the important functions you implemented. Also, write down the header files you created (if any) and What is each header file used for?

- Flow chart and pseudo code: Write the main algorithms used in the program, like game loop, main functionalities, ...
- User Manual: You should make a user manual to teach the user how to use the application.
- Sample runs: You should provide some screenshots showing the program while running through various cases, like winning, playing, losing, ...
- References: Any copied material from anywhere should be referenced. Any discovered unreferenced copied material will result in sever reduction in your grade.

16 Notes

- You are required to work in **groups of two**.
- Take your time to design the project and discuss your design.
- You are required to provide a 3 minute (maximum) video illustrating the problem statement requirements you have fulfilled and any additional bonus features.
- Deliverables should be submitted in Microsoft Teams before the deadline.
- All actual programming should be an independent effort. If any kind of cheating is discovered, penalties will apply to all participating students.
- Start in the project early and come forward with your questions.
- Late submissions are not accepted.
- Prior to discussion, you should be prepared to illustrate your work and answer any questions about it. Failing to do so is a strong indication to copying / cheating.
- It is better to play the actual game so you can understand it better.

Good Luck isA :)