Project 3

Tic Tac Toe

#Details

Duration	15 Days
Submit deadline	15 Oct

You are asked to build a a program that plays Tic Tac Teo with the user.

You are required to draw the grid in the terminal, and ask user for movement then update the grid after make letting the computer plays it turns. And if anyone won, you end the game and show the results and the final grid.

🚨 **NOTEs**:

- For Computer movement, you have the flexibility to do it in any method, starting from most simple and basic ways, to the most complex ways using What is called: Game Theory and Tic Tac Toe Game Theory.
  I Prefer that if you have time to search and try to understand one of them, and if you could apply it in your project, go ahead! ⚔️ 🔥
- It is important to split your project into functions to easy scale it in the next versions. Also learning Header files are important to help you organize your project in better ways.
- It is better to design the logic on paper or any Diagram application so it makes everything easy to build in future. And its important to read and understand all versions so you can imagine how you'll build everything and organize it.
- If you finished building first 3 versions, submit, then if you want you can build the other last 3.

#Tasks

- Build 6 versions, v1, v1.1, v2, v3, v3.1 and v3.1.1

Version 1.0.0.0

- [ ] Build the main logic, where the user starts the first move
- [ ] User moves by selecting an empty position on the grid
- [ ] for every user move, if no winning happened, let the computer to choose an empty place to move to it.
- [ ] if a winning happened, just finish the game and show the result: who won the game and the final grid.
- [ ] Tag this version (v1.0.0)

```
Try to optimize this Game logic as much as you can. You can search for XO Game Theory
```

Version 1.1.0.0

- [ ] For each game, store the final result (Who won), so user can see previous results even if he restarted the program.
- [ ] Before playing, user can choose if he want to play a game, or show the results.

```
When the app runs show 2 options:
[1]: New Game
[2]: History
```

- [ ] When choosing to show results, just show how many times he won and lose

```
LOSES: 10
WINS : 20
```

- [ ] Tag this version (v1.1.0)

Version 2.0.0.0

- [ ] Adding game control:
  - [ ] End game: if the user entered a specific input it ends the game, and doesn't store the results.
  - [ ] Clear results: In the main menu (when the program starts) show a third option called `clear results` that clears all the previous results.

```
When the app runs show 2 options:
[1]: New Game
[2]: History
[3]: Clear History
```

```
When choosing History, show Controls
[0]: Back to main menu
[1]: New Game
[2]: Clear History
```

```
When You are in a new game,
- show the new draw after that move and show this option each time when user makes move:
[- -] Enter: "- -" to end game
```

```
if game is ended in the middle don't save it
after ending, show the first 2 options again
```

- [ ] Tag this version (v2.0.0)

Version 3.0.0.0

- [ ] Create history for each game and store it.
- [ ] Give the user the ability to show see the history for any game.

```
So when user choose to show results, you'll loop on all results like that
game 1: win
game 2: lose
game 3: lose
Total: 1 win, 2 loses

and then asks the user for two options like that:
Enter game ID to represent it or 0 to Back to the main menu:

if he choosed a game, you'll represent the result like that:
X|O|X
O|X|O
O|X|O
```

- [ ] Tag this version (v3.0.0)

Version 3.1.0.0

- [ ] Edit last version to store also date of each game in results. so the history will contain every game details with the date of that game.
- [ ] When the user shows the history, you'll Show results with date like that:

```
game 1 (2024/10/11): win
game 2 (2024/10/11): lose
game 3 (2024/10/12): lose
Total: 1 win, 2 loses
Enter game ID to represent it or 0 to Back to the main menu:
```

- [ ] Tag this version (v3.1.0)

Version 3.1.1.0

- [ ] You must handle old results which has no date recorded, to not cause errors when user shows the old results before the new update. by showing `NULL` instead of the not founded date
- [ ] Tag this version (v3.1.1)

#Conditions-Of-Acceptance

- Do Clean Code.
- Use C++ / C.
- Create the first 3 versions at least with the same tasks mentioned above, and upload all the 3 versions to the same repository on github.
- Last commit of each version should be tagged by the version name.
- Submit before the deadline.
- The Repository should be on your own github account and **public** not private
- AI Generated codes are **NOT** allowed (**Cause you kicked out the camp**)

Finally

The Hashira Corps is waiting for you to join us, so do your best and try as much hard as you can, search and think, imagine and build, and we are excited to see you becoming a Hashira! ⚔️
Do **NOT** hesitate to ask for help if you are stuck, and if you can help others, do it.

This is the way to become #The_Next_Hashira ⚔️