**SODV 1202**

**Object-Oriented Programming - Term Project**

**Project**

**Connect Four**

**Project: Connect Four**

**Overview**

Connect Four is a two-player [connection game](https://en.wikipedia.org/wiki/Connection_game) in which the players first choose a symbol/color and then take turns dropping one Symbol/colored disc from the top into **a seven-column, six-row vertically suspended grid**. The pieces fall straight down, occupying the lowest available space within the column. The objective of the game is to be the first to form a horizontal, vertical, or diagonal line of four of one's own symbols/colored discs–[Wikipedia](https://en.wikipedia.org/wiki/Connect_Four).

**Directions**

The best way to tackle a large project is to break it down into smaller tasks. In this project, you will work **individually** towards the goal of completing the project by completing a series of milestones listed below. The directions for each milestone are listed below. Each milestone must be accomplished/demonstrated in your final project.

**Milestones**

|  |  |  |
| --- | --- | --- |
| 1. | General Strategies | **Work individually**. Use Google/Wikipedia to get an idea of how the connect-4 game works. Brainstorm some general strategies for your project. |
| 2. | Program Skeleton Development | Start developing an object-oriented program to play the Connect Four game. |
| 3. | Setup Git Repository | Prepare your Git repository. Create the development tasks that you have brainstormed. **A GitHub repository link (public) must be sent no later than June 09, 2023.** Use the repository for updating your project code and always create commit statements (Add a caption in the recent code update). |
| 4. | **Send a GitHub link to your instructor** | **In the d2l, final project page, submit the GitHub link and always make frequent commits so that your instructor can inspect your activity and progress. A GitHub link (public) must be sent no later than no later than June 09, 2023. If you do not know how to use the GitHub repository, learn it from YouTube or from online resources.** |
| 5. | Prepare for the future | Throughout the project, you will need to address issues like bugs or new ideas when they are discovered. |
| 6. | Complete Core Classes | **Hints:** **You should design at-least four classes (or more), along with any other classes needed to implement your project.** You may have a controller class to play the game, a model class that implements intermediate steps and holds information about the game, and two classes that extend the Player abstract class. One player can be a human player while the other can be a computer player. Both of these classes should communicate with the world via an object that is created in your controller class. You may also have a class that interacts and provides communication via text input from the keyboard and output on the Console. |
| 7. | Complete Basic 2-user Game | Your primary focus is to complete the game for two human users. Once complete, you can start developing advanced logic for the human vs computer game. |
| 8. | Make sure, OOP principles are followed | **Most of the points will be allocated for understanding and implementing OOP principles, Therefore, before advancing your project further, make sure you have taken advantage of all OOP principles discussed in this course.** |
| 9. | (Optional) Development of an AI algorithm | If time permits, develop an AI algorithm/ rule-based algorithm to make your program compatible with human Vs. computer mode. |
| 10. | **Upload Project source code and video presentation in d2l Dropbox by June 21, 2023, 11:30 PM** | **Your term project final code (source code inside a compressed folder) should be submitted in the D2L Dropbox together with a final project presentation video by the deadline of June 21, 2023, 11:30 PM.** The length of the video presentation could be a maximum of 10 minutes. In the presentation, identify the challenges you have faced, and how you have overcome them. An overall description of how your program works, identify important components and their uses. Run and stop your program multiple times in the demo. |

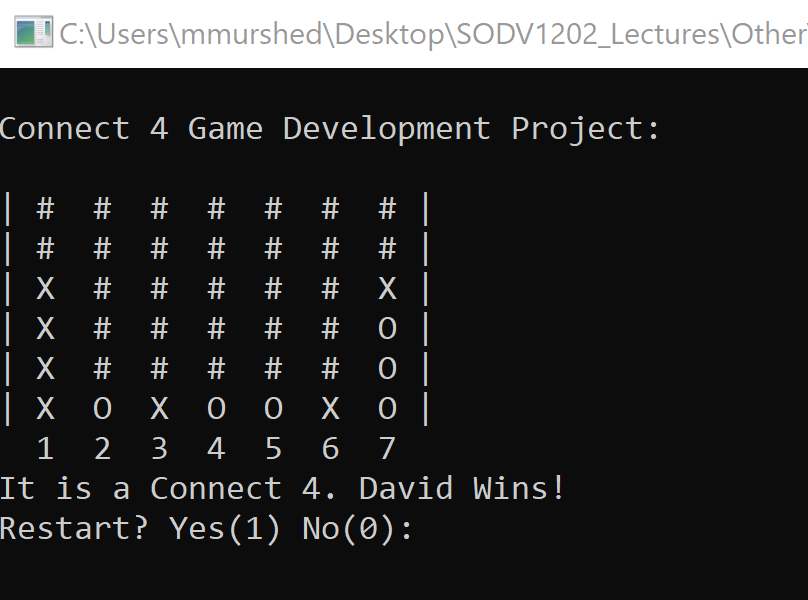
**Rules**

Connect 4 is a classic board game where two players (typically two different colors or two different symbols) take turns dropping disks (symbols) in a grid of **7 columns by 6 rows**. The discs attract gravity and fall to the lowest empty space in their column (also rest on top of other pieces, if any). The winner of the game is the first to have 4 discs of their symbol/color in a row. This can be horizontally, vertically, or diagonally.



Figure: Connect Four.

If you are new to the game, this video (<https://www.youtube.com/watch?v=utXzIFEVPjA>) should help out. It talks about how to play Connect Four. Connect Four is a solved game, meaning, if the first player can make the proper move, he always wins. Therefore, if you can design good algorithms (Optional), and the computer is allowed to make the first move, ideally, the computer should always win. If you want to know the strategies to win connect four (Optional), you can check out this link (<https://www.youtube.com/watch?v=7DnCAgHeFOk>). In your project, you will program Connect 4 game using the numbers as input that has both a 2-player mode and a 1-player mode (Optional). When your program starts, the game should have a start screen where the player can select either 2-player mode or 1-player mode using a console command (Optional).



Snapshot of a console Connect Four Board.

For simplicity, assume the first player is always ‘X’ and the second player is always ‘O’. You can also use text color if you like. Players will interact in the game using numbers. If the first player enters 1, the leftmost column should drop his symbol (‘X’) as deep as possible. However, if the column is already full, the player cannot drop a piece in that column, nothing should happen (your program must not crash here). Once a player has successfully played a piece, his/her turn is over. The game should check if the move created a four-in-a-row or a four-in-a-column or four-in-a-diagonal of the same symbol. If it is the case, your game should print the winner. If not, it should simply pass the turn to the other player. If all 42 blocks in the grid are played, it is a draw, and no one wins.

**(Default mode, must implement this mode)** In the **2-player mode**, players will use numbers to select which column they wish to drop their piece. The game should prompt it is who’s (player’s) turn. The game should clearly indicate which player’s turn it is by using Disc Symbol/ Color/Message (for example, you could have text saying "It is David’s turn:"). The turns should go back and forth until the game ends.

In **1 player mode (Optional)**, the player will play against a computer program. This does not have to be a "good" AI, just some process by which moves are taken automatically without player input. The game should randomly decide to have either the human or the computer go first (since moving first is a big advantage). From there, the game proceeds until it ends.

In either case, when the game ends, the game should show some text indicating either who won or that the game was a draw. The game must then return to the "start" screen, where a player can once again choose either 1-player mode or 2-player mode (Optional).

**Tips**

Do not spend a lot of time working on the AI until (Since it is optional) all basic components are properly functioning using the OOP concept. You can achieve more points by following good programming practices and OOP principles as much as possible. For example, interacting **with multiple class objects, uses of interfaces, adding method overloading, overriding, virtual functions will help you attain more marks** than the AI-based optional design. **Focus on completing the game working first with 2 human players.**

**Always think about OOP concepts!** **Consider what objects to use, how to use properties, values, and attributes to come up with a design with more private methods/functions/properties/fields and a few public methods/functions/properties/fields.**

**Evaluation**

The project will be evaluated for design, code quality, and functionality. A full break-down of how marks will be assigned is listed in the table below.

|  |  |
| --- | --- |
| Task | Marks |
| Design   * The chosen design makes development and maintenance easier * Classes and methods are divided logically * Classes demonstrate with abstraction, property, encapsulation, Inheritance, polymorphism, and good object-oriented design | 5 |
| Effort   * Completed all milestones mentioned in this document * Shared Git-hub repository (public) links, and have made many commit statements. * The project is worked independently. * The coding history/originality of work can be found in the Git-Hub commit statements. * Consistent style coding * Documented with comments inside the program code where necessary * The final project files and the video presentation file are uploaded by the deadline. | 15 |
| Functionality   * Demonstrates that a strategy was planned and implemented. * Functionality must be non-trivial/significant. * The program does not crash | 10 |
| Total | 30 |