|  |  |
| --- | --- |
| **Course Name:** Fundamentals of Programming & Data Science | **Course Code:** CMPE-112L |
| **Assignment Type:** Complex Engineering Problem | **Dated:** 4th-March-2024 |
| **Semester:** 2nd | **Session: 2022** |
| **Lab/Project/Assignment #:** 3 | **CLOs to be covered: CLO 2** |
| **Lab Title:** 3D Tic-Tac-Toe Game | **Teacher Name:** Engr. Afeef Obaid |

**Complex Engineering Problem (CEP):**

|  |  |  |
| --- | --- | --- |
| **No.** | **Attribute** | **Details** |
| **WP1** | Depth of Knowledge required | This problem requires deep knowledge of Python data types, Nested loops and conditional statements. |
| **WP2** | Range of conflicting requirements | This problem involves conflicting requirements, including Complexity vs. Simplicity, Functionality vs. Performance, Error handling, Efficiency and Scalability. Balancing these conflicting requirements will require careful design to make it user friendly. |
| **WP3** | Depth of Analysis  required | This problem required depth of analysis in terms of user interface, validation and choice of efficient data type. |

**CEP Description:**

write a Python program to implement a 3D Tic-Tac-Toe game. The objective is to create a fully functional game that allows two players to play against each other on a 3x3x3 grid.

**Game Implementation:** Game Implementation must include:

**1)** Setting up the 3x3x3 game board.

**2)** Allowing players to take turns placing their symbols ('X' and 'O') on the board.

**3)** Implementing the logic to check for winning conditions in three dimensions.

**4)** Handling draws when the board is full without a winner.

**5)** Providing an option to restart the game.

**User Interface and Input Handling:** Design a user-friendly interface for the game. This could include:

**1)** Displaying the 3D game board visually, so players can easily see the current state of the game.

**2)** Providing prompts and instructions for players to make their moves.

**3)** Clear indication of whose turn it is.

**4)** Displaying messages for game outcomes (win, draw, restart).

**5)** Validating user input to ensure that moves are within the bounds of the board and in unoccupied cells.

**Testing:** Test your implementation rigorously to ensure that it works correctly in all scenarios.

**1)** Winning conditions in all directions (horizontal, vertical, diagonal, and three-dimensional).

**2)** Draws when the board is full.

**3)** Correct behavior when restarting the game.