Completed Requirements

Requirements	Completion
1) Human vs Human	Completed
2) Computer vs Human	Completed
3) Check the validity of moves for players	Completed
4) System can randomly select moves for computer	Completed
5) Player should be able to save and load game from any state	Completed
6) A game should be replay able from any position after loading from saved file	Completed
7) All moves by both players should be undoable and redoable	Completed
8) A primitive online help system	Completed

Declaration of Contribution

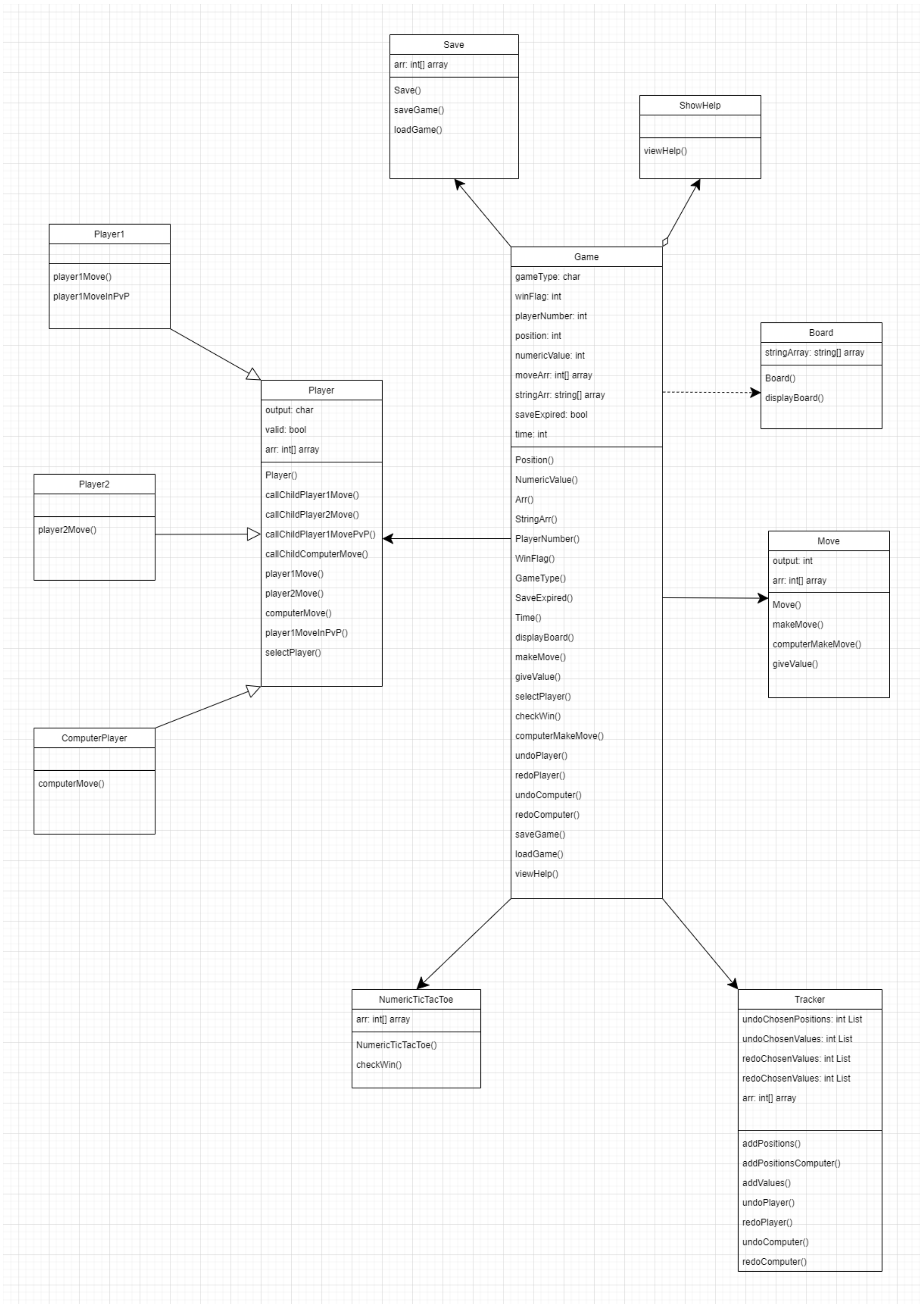
Name	ID	Contribution
Shourav Shome	11392894	100%

Overview of final design

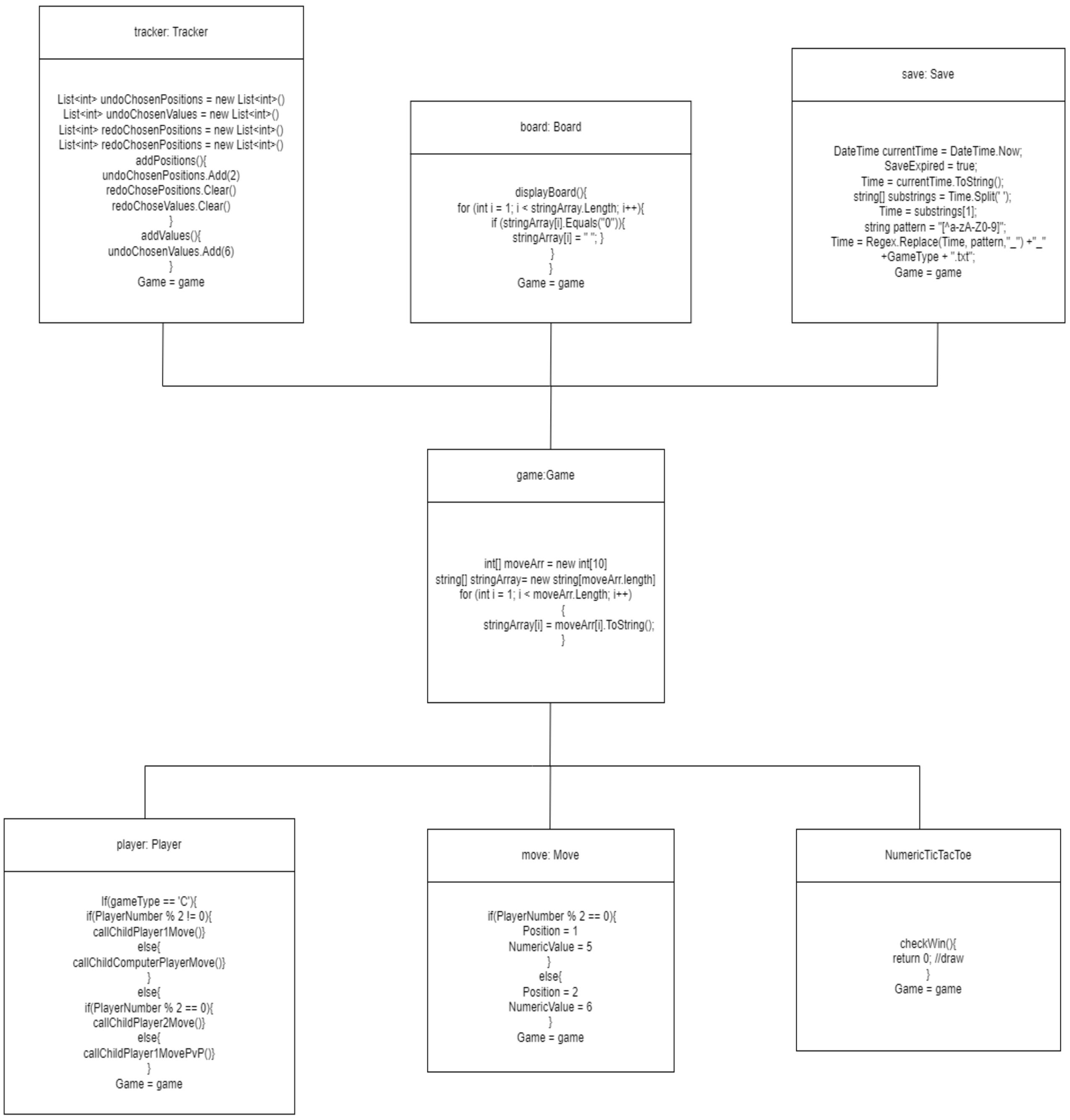
Introduction: This design represents a fully working variant of a Tic Tac Toe game named Numeric Tic Tac Toe. In Numeric Tic Tac Toe, a player has to complete a sum of 15 in any of horizontal, vertical or diagonal line. In my implementation, the first player always begins with odd numbers followed by the second player or computer player with even numbers. All the moves of the game are tracked by an array of size 10. Index 1 corresponds to the first position of the board. So, the index number is used to the represent the positions of the board. At the beginning of the program system asks the player to Start a new game, see help or load game from previously saved files. Player can start a new game and after that player can either give ‘C’ or ‘P’ to start the game. C corresponds to human vs computer and P corresponds to Human vs Human. After selecting the game mode, Player 1 picks a position by giving the values 1 to 9 and the system verifies if the position has any previous values. Before letting the player choose any position, system calls checkWinner function from Numeric Tic Tac Toc class and gets a return of an integer value. If the value is 0 it means the board still has unplayed positions. So, the game continues. If the value is 1 then any of the player has won, the game and if the value is 0 there’s no playable moves remaining and the match is draw. After player 1 selects a valid position, the position is added to undo position list which holds all the positions selected by both the players. Then the system asks the player to give any odd numeric value to put in his selected position. Then this numeric value is added to undo value list. This list is used as tracker for all the moves and values given by the players. Any time player can input ‘Save’ command and the system calls Save class and saveGame function to save the current array. And at the beginning of the program execution, player can load the state of any game from the saved files. For undo and redo functionality 4 lists are used to hold the position and the values given by the players. When a player undoes a move the position and the numeric value is removed from the list and that value is added to the redo list. And when a player redoes a movement, the values are removed from the redo list and added back to undo list.

Reason to change the preliminary design: In my preliminary design I did not have clear idea about designing the tracker class which keeps track of the moves of players. So, in my final design the tracker class is changed. Also, in my previous design there was no separate class to use for determining the moves done by the player. But, in my final design I have introduced a Move class which is called when any player wants to make a move in the game. The winner checking decision is now done by another class name Numeric Tic Tac Toe. This class has a function which returns integer values. Each value represents the current status of the game as if the game is still running or a winner is found or draw. There are few small changes in abstract class Game as well. These changes were made to increase the reusability of the code rather than writing same code again and again. After making these few improvements the whole design has now become much more simplified and easier to understand.

Class Diagram



Object Diagram



```
List<int> undoChosenPositions = new List<int>()
List<int> undoChosenValues = new List<int>()
List<int> redoChosenPositions = new List<int>()
List<int> redoChosenPositions = new List<int>()
addPositions(){
undoChosenPositions.Add(2)
redoChosePositions.Clear()
redoChoseValues.Clear()
}
addValues(){
undoChosenValues.Add(6)
}
Game = game
```

```
displayBoard(){
for (int i = 1; i < stringArray.Length; i++){
if (stringArray[i].Equals("0")){
stringArray[i] = " "; }
}
}
Game = game
```

```
DateTime currentTime = DateTime.Now;
SaveExpired = true;
Time = currentTime.ToString();
string[] substrings = Time.Split(' ');
Time = substrings[1];
string pattern = "[^a-zA-Z0-9]";
Time = Regex.Replace(Time, pattern,"_") +"_"
+GameType + ".txt";
Game = game
```

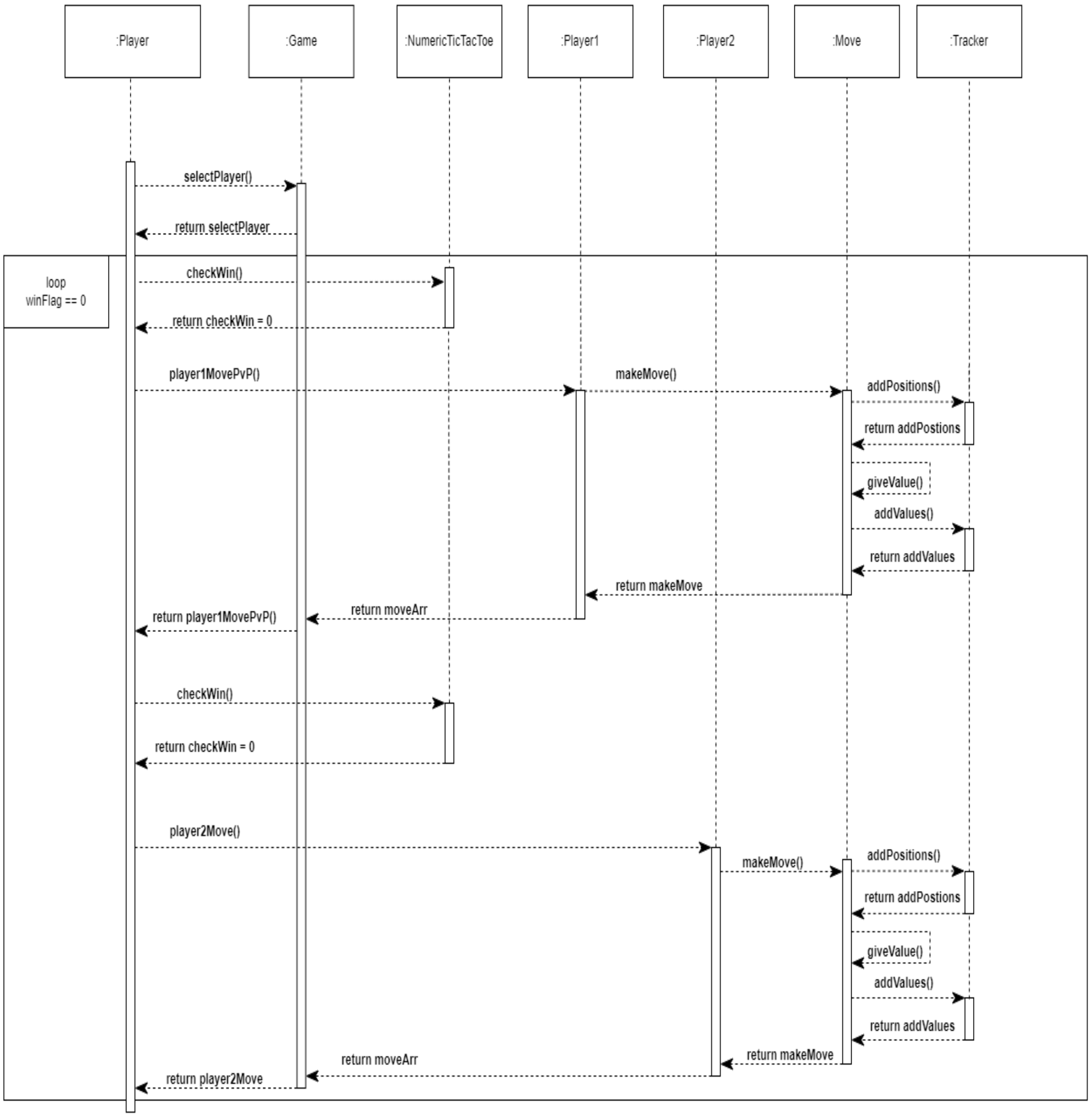
```
int[] moveArr = new int[10]
string[] stringArray= new string[moveArr.length]
for (int i = 1; i < moveArr.Length; i++)
{
        stringArray[i] = moveArr[i].ToString();
}
```

```
If(gameType == 'C'){
if(PlayerNumber % 2 != 0){
callChildPlayer1Move()}
else{
callChildComputerPlayerMove()}
}
else{
if(PlayerNumber % 2 == 0){
callChildPlayer2Move()}
else{
callChildPlayer1MovePvP()}
}
Game = game
```

```
if(PlayerNumber % 2 == 0){
Position = 1
NumericValue = 5
}
else{
Position = 2
NumericValue = 6
}
Game = game
```

```
checkWin(){
return 0; //draw
}
Game = game
```

Sequence Diagram



Design Pattern

The design pattern I followed while implementing this design is template method. Template method is like a skeleton in the superclass which allows the subclasses to override its specific functions and without changing the basic structure but implement new features for the system. The main idea behind template method is to provide a basic structure and let the subclasses provide some of its own implementation. There are few advantages of using template method. Some of the advantages are described in few sentences below:

1. Reusability: By using template method common methods can be used by multiple subclasses. As the methods of the superclass can be overridden anyone can avoid duplicating code which makes the code more maintainable.
2. Flexibility: developers can easily modify existing classes without changing the basic algorithm or the structure of the system. This can be really helpful for the systems which needs constant maintenance and upgradation.
3. Simplified: As a predefined structure exists in this method it is easier to read and understand the whole design and work on specific steps without effecting the whole system.
4. Encapsulation and Abstraction: Template method encapsulates the basic structure and methods inside the superclass so it provides an abstraction layer which can provide good security in term of accessing the skeleton algorithm of the whole system.

Reason behind choosing design pattern: The fast and foremost reason behind choosing template method that this game system can be used for any kind of 2 player traditional board game. All the necessary methods are inside the abstract superclass of the system and all the subclasses can override the necessary methods and provide its own implementation. This design pattern increases the reusability of the system and also provide other developers to understand how the system works. All the Superclasses and subclasses used in this system are stated below with short description.

Game: This class is used as the main abstract superclass of the whole system. The basic algorithm and methods are written here as abstract, and they are later overridden by other subclasses of the system. The main array to hold the positions and values of the players are stored in this Game class.

Move: This class is used to take inputs from the user to take input of the positions and values from the players. makeMove() function is used to take the value of the position from the player and giveValue() is used to take the numeric value for that position. The necessary validation check is implemented inside the method. computerMakeMove() is used to take make moves for the computer player.

Player: Player class is used to determine the player turns and call necessary subclass from Player1, Player2 or ComputerPlayer.

Tracker: Tracker class is used to keep track of the moves of the players and call function when user want to undo or redo a certain move. 4 integer lists are used to hold the positions and values given by the user. Whenever a move is made both the positions and values are added to undoChosenPositions and undoChosenValues list. After that, both the redoChosenPositions and redoChosenValues list is reset to default. When undoPlayer() method is called the positions and values are removed from undo lists and added to redo lists. Again when redoPlayer() is called values from redo list is removed and added back to undo list.

Save: Save subclass holds the saveGame() and loadGame() functions to save a game and load a game at any state. Players can save a game at any state of the game, but players can only load a saved game at the beginning of the execution of the program.

ShowHelp: Players can call this help function by simply selecting 2 at the beginning. The viewHelp() functions holds the instruction example how to play this game.

Program execution: My code was built on .NET6 environment and should run on all .NET6 environment. The program can be executed by running ConsoleApp3.sln. Some variables are used on runtime. Time variable is used to take current time of the system by removing other special characters and converting it to string. That string variable is used as the name of the saved file. This time variable is only taken once in the game so each time the game is saved in the current execution the file is over written. Also, for the undo and redo options if player1 undoes their move it automatically removes the previous value given by the computer. Again redo brings back the player and computers values. Same method is followed for the human vs human format. If player 1 undoes a turn it removes player2's previous turn including players turn and set the player turn to player 1 again.

To execute the program first the solution file has to be run. After successfully compiling the code the console will ask if the player wants to start a new game, load help options or load a game from saved files. After that the system will ask if user wants to start a game human vs computer or human vs human. To select computer vs human player needs to give 'C' and to start player vs player, player needs to give 'P'. After that the system will check for Winflag. If its 0 it will ask player 1 to select a position and after that it will ask the player to give an odd value in that position. The system will validate the given value. Player can also put 'Save' 'Undo1' or 'Redo1' while choosing position to perform an action. After player 1 operation system again check for Winflag and if its 0 player2 gets the turn.

Used Existing Libraries: System.Text.RegularExpressions is used to clear out the special characters from the time variable and then stored in a variable.