Program Description for Developers

Software Development:

- Written in C++
- Compiled in FEH Qt Creator
- Makes use of provided FEHLCD.h, FEHIO.h, FEHUtility.h, and LCDColors.h libraries

Assumptions:

- Players will not tap in a column if it is already full
- Players will not double tap/tap too quickly
- Players know how to read English and can follow instructions

Variables and Uses:

- Int **board[5][5]**
  - Integer Array to represent what tokens are in each space of the board display
  - Empty slots are represented by 0
  - Blue and red token filled slots are represented by 1 and 2, respectively
  - Used to find when a player wins
- Int **n**
  - Stores the winner of a game
  - If n = 1, Blue wins
  - If n = 2, Red wins
  - If n = 3, it's a tie game
  - Set by WinState()
- Float **tapx**
  - stores the x-coordinate on the screen of where the player taps
  - Altered by the Check_Touch() function
- Float **tapy**
  - stores the y-coordinate on the screen of where the player taps
  - Altered by the Check_Touch() function
- Int **menu_choice**
  - Stores which icon the player taps on for use in a switch case to decide what screen to go to
  - Start button = 1, Instructions button = 2
- Int **column_index**
  - Stores which column the player taps on to determine which column to check for open slots in the function Find_Slot()
  - Used in Fill_Slot() to determine the x value of the center of the token to be drawn

- Int **row_index**
    - Stores the lowest open slot in the column chosen by the user
    - Set by the function Find_Slot()
    - Used in Fill_Slot() to determine the x value of the center of the token to be drawn
- Int **x_slot_center[5]**
    - Stores the x coordinates of the centers of each column
    - Set by Slot_Centers()
    - Used by Fill_Slot() for the x-coordinate of the center of the drawn circle
- Int **y_slot_center[5]**
    - Stores the x coordinates of the centers of each row
    - Set by Slot_Centers()
    - Used by Fill_Slot() for the y-coordinate of the center of the drawn circle
- Int **red_win_count**
    - Counts number of wins for the red user
    - Set by WinStates()
    - Used in DisplayStats(), prints how many red wins to screen
- Int **blue_win_count**
    - Counts number of wins for the blue user
    - Set by WinStates()
    - Used in DisplayStats(), prints how many blue wins to screen
- Int **token_type**
    - Stores the color of the token to be drawn for each player
    - 1 = blue, 2 = red
    - Set by PlayerBlue(), PlayerRed()

Functions:

- **Connect4()**
    - Constructor function, initializes variables found in the class for the object, no specific type, no calling parameters. Declare by typing Connect4::Connect4() {arguments}
- **BlankBoard()**
    - Resets graphical board to initial blank tokenless state, type void, no calling parameters. Use by typing objectname.BlankBoard();
- **PlayerBlue()**
    - Registers blue player touch input and converts to blue token placement, type void, no calling parameters. Use by typing objectname.PlayerBlue();
- **PlayerRed()**
    - Registers red player touch input and converts to red token placement, type void, no calling parameters. Use by typing objectname.PlayerRed();
- **WinState()**

    - Analyzes board[5][5] array after each player turn to check if the game has resulted in a blue win, a red win, or a tie; additionally displays winner to screen via colored text. Type void, no calling parameters. Use by typing objectname.Winstate();
- **GameLoop()**
    - Loops for one iteration of the game; contains a loop that includes PlayerBlue(), PlayerRed(), and WinState() that loops until it registers an end of game. Type void, no calling parameters. Use by typing objectname.GameLoop();
- **Menu_Function()**
    - Displays the title menu, and gives touch option for start or instructions pages. Type void, no calling parameters. Use by typing objectname.Menu_Function();
- **Display_Menu()**
    - Displays the graphical component of the title menu. Type void, no calling parameters. Use by typing objectname.Display_Menu();
- **Check_Touch()**
    - Checks if screen has been touched and stores pixel coordinates of tap. Type void, no calling parameters. Use by typing objectname.Check_Touch();
- **Menu_Button_Check()**
    - Checks if pixel coordinates of tap are on start button or are on instructions button. Type void, no calling parameters. Use by typing objectname.Menu_Button_Check();
- **Display_Instructions()**
    - Displays written instructions graphically to screen. Type void, no calling parameters. Use by typing objectname.Display_Instructions();
- **Column_Check()**
    - Checks what pixel coordinates the column tap of the user were in and stores column as an array index. Type void, no calling parameters. Use by typing objectname.Column_Check();
- **Instructions_Button()**
    - Runs what should happen if user presses the instructions button on the title menu. Type void, no calling parameters. Use by typing objectname.Instructions_Button();
- **Slot_Centers()**
    - Calculates the pixel coordinates of the center of each graphical slot on the game board and stores values in an array. Type void, no calling parameters. Use by typing objectname.Slot_Centers();
- **Find_Slot()**
    - Finds the lowest row with an open slot in the user chosen column. Type void, no calling parameters. Use by typing objectname.Find_Slot();
- **Fill_Slot()**
    - Fills the graphical board of the found slot with a circle of the corresponding player's color; changes the board[5][5] array index of the location to mirror graphical board. Type void, no calling parameters. Use by typing objectname.Fill_Slot();
- **BoardClear()**
    - Clears the board[5][5] array to reset it for a future game. Type void, no calling parameters. Use by typing objectname.BoardClear();

- **CreditDisplay()**
    - Displays thank you message and credits to screen after end of game is detected. Type void, no calling parameters. Use by typing objectname.CreditDisplay();
- **DisplayStats()**
    - Displays win count for each player to screen, if user taps again they go back to title menu. Type void, no calling parameters. Use by typing objectname.DisplayStats();

Program Performance and Limitations:

- When a player taps twice by accident, it places a token in the column tapped in, thus stealing the other player's turn
- When a player taps in a full column, the top token is overwritten to whatever the player's color is
- Statistics reset after turning off the Proteus
- Players have to wait through the credits screen after playing before they can go back to the menu