3. WRITTEN RESPONSES

3 a.

3.a.i.

The overall purpose of the program is to facilitate a "connect-four" game play experience between two players and to detect a winner (four pieces in a row).

3.a.ii.

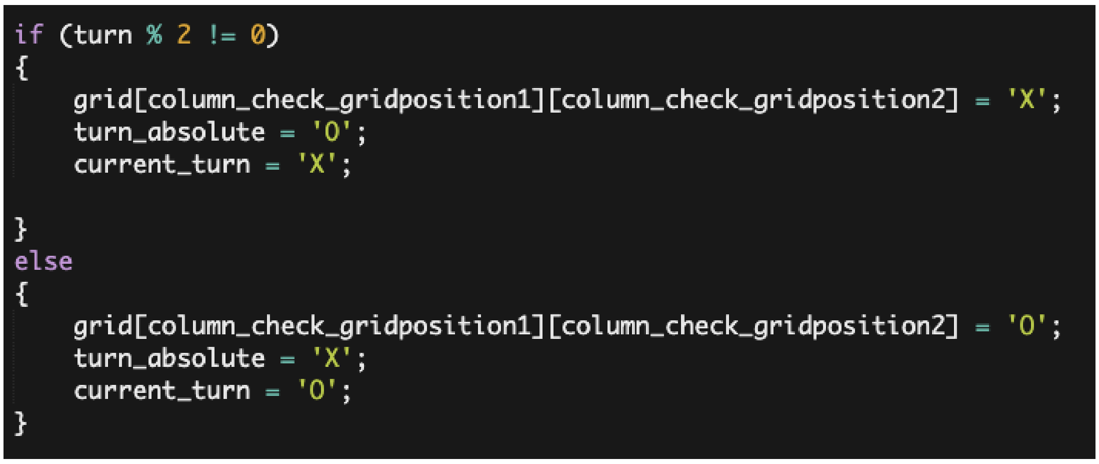
The video demonstrates simulated gameplay between two players: Player X and Player O. In the video, the program prompts Player X to type the column they'd like to play in, and re-prompts the player until they enter an accepted value. The program then displays the updated game board. This process repeats until the program detects that a player has achieved victory (four pieces in a row), at which point the program is terminated. The video demonstrates the ability of the program to detect all win scenarios.

3.a.iii.

In the video, numerous inputs are given by Player X and Player O, corresponding to the desired column they'd like to place their piece. The program then prints the updated game board and prompts the next player for input. If a player achieves four-in-a-row, the program additionally outputs a statement identifying the winner then terminates.

3 b.

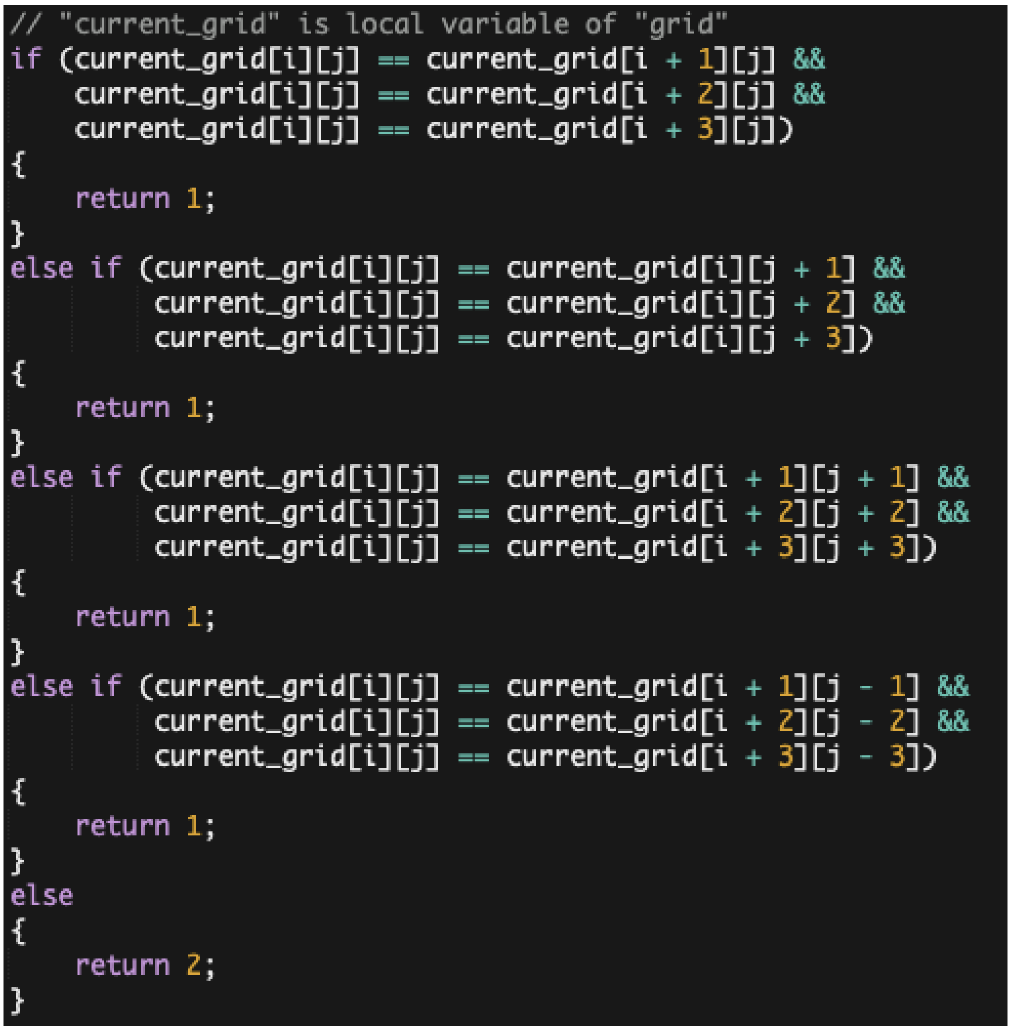
3.b.i.



```
if (turn % 2 != 0)
{
    grid[column_check_gridposition1][column_check_gridposition2] = 'X';
    turn_absolute = 'O';
    current_turn = 'X';

}
else
{
    grid[column_check_gridposition1][column_check_gridposition2] = 'O';
    turn_absolute = 'X';
    current_turn = 'O';
}
```

3.b.ii.



```
// "current_grid" is local variable of "grid"
if (current_grid[i][j] == current_grid[i + 1][j] &&
    current_grid[i][j] == current_grid[i + 2][j] &&
    current_grid[i][j] == current_grid[i + 3][j])
{
    return 1;
}
else if (current_grid[i][j] == current_grid[i][j + 1] &&
         current_grid[i][j] == current_grid[i][j + 2] &&
         current_grid[i][j] == current_grid[i][j + 3])
{
    return 1;
}
else if (current_grid[i][j] == current_grid[i + 1][j + 1] &&
         current_grid[i][j] == current_grid[i + 2][j + 2] &&
         current_grid[i][j] == current_grid[i + 3][j + 3])
{
    return 1;
}
else if (current_grid[i][j] == current_grid[i + 1][j - 1] &&
         current_grid[i][j] == current_grid[i + 2][j - 2] &&
         current_grid[i][j] == current_grid[i + 3][j - 3])
{
    return 1;
}
else
{
    return 2;
}
```

3.b.iii.

The list (2-dimensional array) used in the code above is given the name "grid." In the second image, "grid" is accessed within a function as an argument and given the local name "current_grid."

3.b.iv.

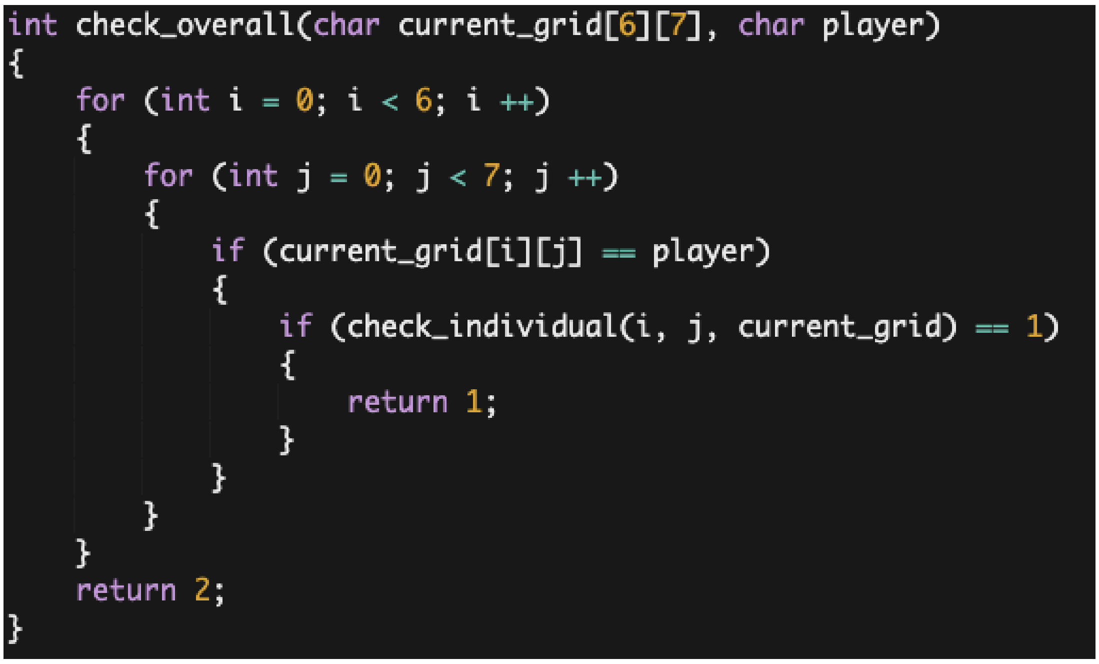
The data contained in this list represents the "status" of each position on the connect-four game board. The array is initialized with the dimensions of 6 x 7, with each position representing the respective position on the game board. The "status" is denoted by an "X" (piece belongs to player X), an "O" (piece belongs to player O), or a " " (position is empty).

3.b.v.

Without the list, the program would have no way to remember previous player inputs, making gameplay impossible. The list allows for the program to save and update the status of every position on the game board for every turn. In turn, the saved data in the list can then be accessed later, enabling other important functions such as printing the updated game board after every turn (print_grid). The list is also accessed to determine which position is to be filled next in each column. Additionally, after every turn, the list is accessed in the check_overall and check_individual functions to determine whether a player has won.

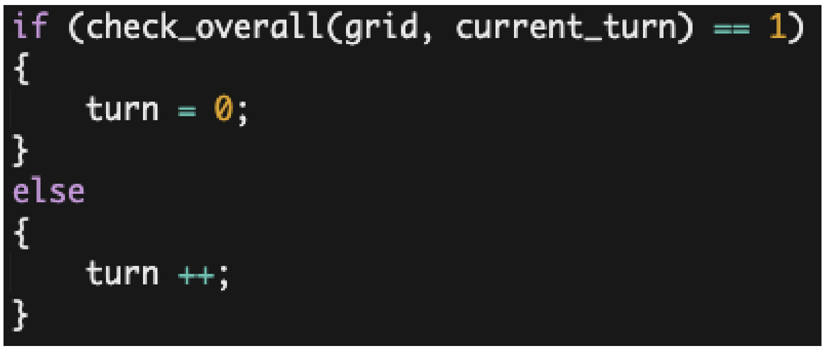
3 c.

3.c.i.



```
int check_overall(char current_grid[6][7], char player)
{
    for (int i = 0; i < 6; i ++)
    {
        for (int j = 0; j < 7; j ++)
        {
            if (current_grid[i][j] == player)
            {
                if (check_individual(i, j, current_grid) == 1)
                {
                    return 1;
                }
            }
        }
    }
    return 2;
}
```

3.c.ii.



```
if (check_overall(grid, current_turn) == 1)
{
    turn = 0;
}
else
{
    turn ++;
}
```

3.c.iii.

After each player's turn, this function iterates through each item in the array (each position on the game board) to determine if a row of four like-pieces stems from it. This contributes to the overall functionality of the program by checking after every player's turn to see if they've won.

3.c.iv.

The function (check_overall) is given an argument "current_grid" which is the updated array "grid" after each players turn and the argument "player" which is the player whose turn it was. It then works by using a "for loop" that runs six times, then nesting another "for loop" inside that runs seven times. The outer loop will initialize the variable "i" as 0 and increment by +1 each time it runs, while the inner loop will initialize the variable "j" as 0 and increment by +1 each time it runs. In the inner loop, If the piece belongs to "player", check_overall will call the check_individual function and pass in "i," "j," and "current_grid" as arguments. This function will access the item current_grid[i][j] and will determine whether a vertical, horizontal, or diagonal sequence of four pieces of the same owner stems from that position. If it does determine such a sequence, check_individual will return 1 and save it into the "status" variable. Otherwise, check_individual will return 2 and save into "status." As check_overall iterates through all forty-two positions, if "status" is ever 1, the function will return 1 and exit the check_overall function. If "status" never equals 1 after iterating through every position, check_overall will return 2.

3 d.

3.d.i.

First call:

The function is called after the first turn when the array "grid" has been updated so that all positions have the status " " except for one position which has status "X". "grid" is then passed into check_overall as argument "current_grid" and "X" is passed as argument "player."

Second call:

In another instance where the function is called, player input has updated the array "grid" such that Player O has four "O" pieces in an uninterrupted vertical line. The status of all other pieces are determined by previous player inputs such that no other "connect-four's" exist. This new "grid" is then passed into check_overall and "O" is passed as argument "player."

3 d.ii.

Condition(s) tested by first call:

The first call tests whether four positions in a row of any kind (horizontal, etc.) are occupied by "X."

Condition(s) tested by second call:

The second call tests whether four positions in a row of any kind are occupied by "O."

3.d.iii.

Results of the first call:

In the first call, due to Player X not having a four-in-a-row, the function returns 2 which does not satisfy the "if" statement in the main function and enters the "else" protocol, proceeding to the next player's turn.

Results of the second call:

In the second call, the function identifies the existence of four-in-a-row for Player O and returns 1 which satisfies the "if" statement in the function. This sets the "turn" variable to 0, breaking the loop and ending the game.