Mariana Reyes

Tic-Tac-Toe
Assignment 1

Task: 1 & 2

tictactoe.c

```c
#include <stdio.h>

char game_grid[3][3] = {
      {' ', ' ', ' '},
      {' ', ' ', ' '},
      {' ', ' ', ' '}
};

// Function to save the current game state to a file
void saveGameState() {
   FILE *fp;
   fp = fopen("game_state.txt", "w");
   if (fp == NULL) {
       printf("Error opening file!\n");
       return;
   }

   for (int i = 0; i < 3; i++) {
       for (int j = 0; j < 3; j++) {
           fprintf(fp, "%c", game_grid[i][j]);
           if (j < 2) {
               fprintf(fp, " ");
           }
       }
       fprintf(fp, "\n");
   }

   fclose(fp);
   printf("Game state saved successfully.\n");
}

void loadGameState() {
   FILE *fp;
   fp = fopen("game_state.txt", "r");
   if (fp == NULL) {
       printf("No saved game state found.\n");
       return;
   }

   for (int i = 0; i < 3; i++) {
       for (int j = 0; j < 3; j++) {
           fscanf(fp, " %c", &game_grid[i][j]);
           fseek(fp, 1, SEEK_CUR);  // Move the file pointer to skip the space character
       }
   }

   fclose(fp);
   printf("Game state loaded successfully.\n");
}

//Display grid in current state
void displayGrid(char game_grid[3][3]){
   //Do a for loop to iterate through the matrix
   for(int i = 0; i < 3; i++){
       for(int j = 0; j < 3; j++){
           printf("%c", game_grid[i][j]);
           if (j < 2) {
               printf(" | "); // Add vertical line
           }
       }
       printf("%s", "\n");
       if (i < 2) {
```

```
            printf("---------\n"); // Add horizontal lines
        }
    }
    printf("%s", "\n");
}

//Checking if a player wins the game
int checkWin(char game_grid[3][3], char player){
    //8 possible ways of winning
    for (int i = 0; i < 3; i++) {
        if (game_grid[i][0] == player && game_grid[i][1] == player && game_grid[i][2] == player) {
            return 1; // Row win
        }
        if (game_grid[0][i] == player && game_grid[1][i] == player && game_grid[2][i] == player) {
            return 1; // Column win
        }
    }
    if (game_grid[0][0] == player && game_grid[1][1] == player && game_grid[2][2] == player) {
        return 1; // Diagonal win (top-left to bottom-right)
    }
    if (game_grid[0][2] == player && game_grid[1][1] == player && game_grid[2][0] == player) {
        return 1; // Diagonal win (top-right to bottom-left)
    }
    return 0; // No win
}

//Check if inputs provided by the user are valid
int isValidMove(char game_grid[3][3], int choice){
    int row = (choice - 1) / 3;
    int column = (choice - 1) % 3;
    if (choice >= 1 && choice <= 9 && game_grid[row][column] == ' ') {
        return 1; // Valid move
    } else {
        return 0; // Invalid move
    }
}

//Check if the game is a draw
int isDraw(char game_grid[3][3]){
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (game_grid[i][j] == ' ') {
                return 0; // Game is not a draw
            }
        }
    }
    return 1; // Game is a draw
}

void play() {
    //Defining the players
    char player_x = 'x';
    char player_o = 'o';
    int turns = 2;
    int choice;

    char currentPlayer = 'x';

    //Making moves until one of them wins, or it is a draw
    while(checkWin(game_grid, player_x) == 0 && checkWin(game_grid, player_o) == 0 && isDraw(game_grid) == 0){
        if(turns % 2 == 0){ // x plays
            printf("%s", "Where do you want to play your x (1-9)?\n");
            printf("%s", "1. Top left \n2. Top middle \n3. Top right \n4. Middle left \n5. Middle \n6. Middle
right\n7. Bottom left \n8. Bottom middle \n9. Bottom rigth\n");
            scanf("%d", &choice);
            //Check if it is a valid move
            if(isValidMove(game_grid, choice) == 1){
                int row = (choice - 1) / 3;
                int column = (choice - 1) % 3;
                game_grid[row][column] = player_x;
                turns++;
            } else{
                printf("%s", "Invalid move, try again!\n");
            }
        } else { //o plays
            printf("%s", "Where do you want to play your o (1-9)?\n");
```

```
            printf("%s", "1. Top left \n2. Top middle \n3. Top right \n4. Middle left \n5. Middle \n6. Middle
right\n7. Bottom left \n8. Bottom middle \n9. Bottom rigth\n");
            scanf("%d", &choice);
            //Check if it is a valid move
            if(isValidMove(game_grid, choice) == 1){
                int row = (choice - 1) / 3;
                int column = (choice - 1) % 3;
                game_grid[row][column] = player_o;
                turns++;
            } else{
                printf("%s", "Invalid move, try again!\n");
            }
        }
        displayGrid(game_grid);

        // Save the game state after each move if the game is not won
        if (checkWin(game_grid, player_x) == 0 && checkWin(game_grid, player_o) == 0 && isDraw(game_grid) == 0)
{
            saveGameState();
        }
    }

    // Determine the winner or if it's a draw
    if (checkWin(game_grid, player_x)) {
        printf("Player X wins!\n");
    } else if (checkWin(game_grid, player_o)) {
        printf("Player O wins!\n");
    } else {
        printf("It's a draw!\n");
    }

    displayGrid(game_grid);
}

int main(void){
    int choice;
    printf("Do you want to reload a previous game? (1 for yes)\n");
    scanf("%d", &choice);

    if (choice == 1) {
        loadGameState();
        displayGrid(game_grid);
    }

    play();

    return 0;
}
```

Task: 3

test_tictactoe.c

```
#include <stdio.h>
#include <CUnit/Basic.h>

//Display grid in current state
void displayGrid(char game_grid[3][3]){
    //Do a for loop to iterate through the matrix
    for(int i = 0; i < 3; i++){
        for(int j = 0; j < 3; j++){
            printf("%c", game_grid[i][j]);
            if (j < 2) {
                printf(" | "); // Add vertical line
            }
        }
        printf("%s", "\n");
        if (i < 2) {
            printf("---------\n"); // Add horizontal lines
        }
    }
```

```
   printf("%s", "\n");
}

//Checking if a player wins the game
int checkWin(char game_grid[3][3], char player){
   //8 possible ways of winning
   for (int i = 0; i < 3; i++) {
       if (game_grid[i][0] == player && game_grid[i][1] == player && game_grid[i][2] == player) {
           return 1; // Row win
       }
       if (game_grid[0][i] == player && game_grid[1][i] == player && game_grid[2][i] == player) {
           return 1; // Column win
       }
   }
   if (game_grid[0][0] == player && game_grid[1][1] == player && game_grid[2][2] == player) {
       return 1; // Diagonal win (top-left to bottom-right)
   }
   if (game_grid[0][2] == player && game_grid[1][1] == player && game_grid[2][0] == player) {
       return 1; // Diagonal win (top-right to bottom-left)
   }
   return 0; // No win
}

//Check if inputs provided by the user are valid
int isValidMove(char game_grid[3][3], int choice){
   int row = (choice - 1) / 3;
   int column = (choice - 1) % 3;
   if (choice >= 1 && choice <= 9 && game_grid[row][column] == ' ') {
       return 1; // Valid move
   } else {
       return 0; // Invalid move
   }
}

//Check if the game is a draw
int isDraw(char game_grid[3][3]){
   for (int i = 0; i < 3; i++) {
       for (int j = 0; j < 3; j++) {
           if (game_grid[i][j] == ' ') {
               return 0; // Game is not a draw
           }
       }
   }
   return 1; // Game is a draw
}

int init_suite(void) {
   return 0;
}

int clean_suite(void) {
   return 0;
}

void test_isValidMove(void) {
 char game_grid[3][3] = {
       {' ', ' ', ' '},
      {' ', ' ', ' '},
        {' ', ' ', ' '}
 };

 CU_ASSERT(isValidMove(game_grid, 5) == 1); // Valid move
 CU_ASSERT(isValidMove(game_grid, 10) == 0); // Invalid move
}

void test_checkWin(void) {
   char game_grid[3][3] = {
           {'x', 'o', 'x'},
           {' ', 'x', 'o'},
           {'o', ' ', 'x'}
   };

   CU_ASSERT(checkWin(game_grid, 'x') == 1); // X wins
   CU_ASSERT(checkWin(game_grid, 'o') == 0); // O doesn't win
   // Add more test cases as needed
}
```

```c
void test_isDraw(void) {
   char game_grid[3][3] = {
           {'x', 'o', 'x'},
           {'x', 'x', 'o'},
           {'o', 'x', 'o'}
   };

   CU_ASSERT(isDraw(game_grid) == 1); // draw
   // Add more test cases as needed
}

int main() {
   CU_pSuite pSuite = NULL;

   if (CUE_SUCCESS != CU_initialize_registry()) {
       return CU_get_error();
   }

   pSuite = CU_add_suite("TicTacToe_Test_Suite", init_suite, clean_suite);
   if (NULL == pSuite) {
       CU_cleanup_registry();
       return CU_get_error();
   }

   //(NULL == CU_add_test(pSuite, "test of isValidMove", test_isValidMove)) ||
   if ((NULL == CU_add_test(pSuite, "test of isValidMove", test_isValidMove)) ||
   (NULL == CU_add_test(pSuite, "test of checkWin", test_checkWin)) ||
   (NULL == CU_add_test(pSuite, "test of isDraw", test_isDraw))) {
       CU_cleanup_registry();
       return CU_get_error();
   }

   CU_basic_set_mode(CU_BRM_VERBOSE);
   CU_basic_run_tests();
   CU_cleanup_registry();

   return CU_get_error();
}
```

Test result after: The results did not help me uncover some issues. My code works perfectly.

*[master][~/CLionProjects/CProgramming]$ gcc test_tictactoe.c -o ttt2 -lcunit
*[master][~/CLionProjects/CProgramming]$ ./ttt2

```
[*[master][~/CLionProjects/CProgramming]$ gcc test_tictactoe.c -o ttt2 -lcunit
[*[master][~/CLionProjects/CProgramming]$ ./ttt2


     CUnit - A unit testing framework for C - Version 2.1-3
     http://cunit.sourceforge.net/


Suite: TicTacToe_Test_Suite
  Test: test of isValidMove ...passed
  Test: test of checkWin ...passed
  Test: test of isDraw ...passed

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      1      1    n/a      0        0
               tests      3      3      3      0        0
             asserts      5      5      5      0      n/a

Elapsed time =    0.000 seconds
```

Task: 4

Debugging

Before winning:

```
(lldb) b tictactoe.c:122
Breakpoint 3: where = crash`play + 27 at tictactoe.c:122:11, address = 0x0000000100003a0b
```

```
(lldb) b tictactoe.c:159
Breakpoint 4: where = crash`play + 613 at tictactoe.c:159:9, address = 0x0000000100003c55
(lldb) run
There is a running process, kill it and restart?: [Y/n] y
Process 67326 exited with status = 9 (0x00000009)
Process 67359 launched: '/Users/marianareyes/CLionProjects/CProgramming/crash' (x86_64)
Do you want to reload a previous game? (1 for yes)
2
Process 67359 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 3.1
    frame #0: 0x0000000100003a0b crash`play at tictactoe.c:122:11
   119      char currentPlayer = 'x';
   120
   121      //Making moves until one of them wins, or it is a draw
-> 122      while(checkWin(game_grid, player_x) == 0 && checkWin(game_grid, player_o) == 0 && isDraw(game_grid) == 0){
   123          if(turns % 2 == 0){ // x plays
   124              printf("%s", "Where do you want to play your x (1-9)?\n");
   125              printf("%s", "1. Top left \n2. Top middle \n3. Top right \n4. Middle left \n5. Middle \n6. Middle right\n7. Bottom left \n8. Bottom middle \n9. Bottom rigth\n");
Target 0: (crash) stopped.
```

Before game finishes with a draw:

```
Process 67586 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 1.1
    frame #0: 0x0000000100003c55 crash2`play at tictactoe.c:159:9
   156      }
   157
   158      // Determine the winner or if it's a draw
-> 159      if (checkWin(game_grid, player_x)) {
   160          printf("Player X wins!\n");
   161      } else if (checkWin(game_grid, player_o)) {
   162          printf("Player O wins!\n");
Target 0: (crash2) stopped.
```

Explanation: First it tells where the process stopped to inspect the state of the running program, 67359 for before winning and 67586 for before ending with a draw. The reason for stopping both are breakpoints that were manually placed. Then it tells the memory address before execution, 0x0000000100003a0b for before winning and 0x0000000100003c55 before ending with a draw. It later shows you the place in your program where you set the break and stops the code in general. **(Below the scenarios are printed)**

```
Process 69077 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 1.1
    frame #0: 0x0000000100003c55 crash4`play at tictactoe.c:159:9
   156      }
   157
   158      // Determine the winner or if it's a draw
-> 159      if (checkWin(game_grid, player_x)) {
   160          printf("Player X wins!\n");
   161      } else if (checkWin(game_grid, player_o)) {
   162          printf("Player O wins!\n");
Target 0: (crash4) stopped.
(lldb) print game_grid
(char[3][3]) $0 = ["xox","oxo","x  "]
(lldb) c
Process 69077 resuming
Player X wins!
x | o | x
---------
o | x | o
---------
x |   |

Process 69077 exited with status = 0 (0x00000000)
```

```
Process 69311 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 1.1
    frame #0: 0x0000000100003c55 crash4`play at tictactoe.c:159:9
   156      }
   157
   158      // Determine the winner or if it's a draw
-> 159      if (checkWin(game_grid, player_x)) {
   160          printf("Player X wins!\n");
   161      } else if (checkWin(game_grid, player_o)) {
   162          printf("Player O wins!\n");
Target 0: (crash4) stopped.
(lldb) c
Process 69311 resuming
It's a draw!
x | o | x
---------
o | o | x
---------
x | x | o

Process 69311 exited with status = 0 (0x00000000)
(lldb)
```