Project 3 and 4 consisted of 4 games where the player chooses which game to play from a main menu. For this code review we decided to choose an informal walkthrough as our code review platform since the games are terminal based and it would be easy to split up the tasks if problems arise. Each paragraph walks through their respective game code, first paragraph is tic tac toe, second is Hangman, trivia is third and Blackjack the fourth paragraph.

Tic tac Toe consists of a .h and .cpp file
The main complaint was how the winning conditions were originally implemented.
The person who wrote the test files decided to place this in an independent function and a small code refactor fixed this problem.
This change made it easier to test and made the code easier to read.

Hangman consists of a .h and .cpp file. This program also utilizes a .txt file which the program reads from. When the Hangman program first released, The lead designer implemented words that the program randomly chose from a pool of 10 strings.
Claire made the program read from a text file with over 3,000 words which randomly chooses one of the words to be a guess word. Another problem that was brought up is the use of capital letters vs lowercase. This problem has been flagged and documented, the approach to this is more of a "soft solution" rather than a code implementation. Our solution was to implement rules for the game and mention that all guess' must be lowercase. This framework has been implemented for all the games which require input.

Trivia consists of trivia.h and trivia.cpp
Originally this game would have had answers ranging from one to two words. This game had problems implementing two word answers. Parveen Kaur implemented the original game and addressed the issue. The solution was to eliminate the questions which could lead to multiple word answers. Those questions were replaced with new single word answers. Now, all the questions only require one word. The problem that affected TicTacToe also was present in trivia where the code could not be tested easily. Parveen refactored and cleaned the codebase considerably. This allowed for easier testing and readability, the lines of code from the cpp file went from ~700 down to ~300.

BlackJack consists of Player.h Player.cpp and both executive_BlackJack.h and executive_BlackJack.cpp files. BlackJack had problems when using a stack to store values. The Original intent was to pop the values and showcase the cards one by one on the terminal. Eduardo is the designer of this game and made the decision to revert to using a member variable method. A member variable that would store the values in the Player class. This also drove the decision to show the players card in a different way than what was previously mentioned. Another change was the implementation of multiple players to participate in this game. However to keep with the theme of single player games, this was reverted back to only one player.