Within the game of checkers many aspects of the game need to be tested to ensure that the game is functioning correctly. If tests are not made to test these aspects, a possible bug could be discovered which could break the functionality of the game, or limit how the user can interact with the game.

The first and arguably most important aspect of the game would be that players can move and are given the correct targets when a piece is selected. This is addressed in the testBasicTargetSelection() function. To ensure that pieces are selecting the correct targets and moving the correct places, both the upper right and lower left corners are tested. A variety of other tests involving all four sides of the board, as well as moving when on the opponents side of the board. All these test ensure that the pieces will move as intended and do not move to an occupied square.

testBasicKingTargetSelection() { is the next suite of tests which parallel the BasicTargetSelection. In checkers the movement of kings becomes a very important to finishing a game and capturing remaining pieces. The tests conclude that border cases as well as normal movement can be made by kinged pieces.

testAdvancedTargetSelection() uses the basic piece and sets the board so that jumps can be made over one piece, two pieces, as well as multiple jump locations. It also makes an important assertion that a piece cannot jump over another of the same color. This is a crucial to the game of checkers as without this functionality the game cannot be completed as no opponent’s pieces can be captured.

testAdvancedKingTargetSelection() is very similar to the normal Advanced selection but adds the test of a backwards function as kings are allowed to make jumps to the side of their color. Otherwise it tests the same functionality of the previous tests.

testRemove() checks if pieces are removed correctly. As a piece is jumped the function remove will be called to remove the jumped piece. This allows for the captures of your opponents pieces. This tests sets the board with pieces then removes them. If this function is working correctly, a jump can be performed correctly and effect the board in the correct manner.

testMove() uses the checkLocation() function to move the pieces once the correct targets have been found. Within this test, the locations of moved pieces is verified that they are correct. It is also important to test when a piece becomes a king, so we bundled the movement of the pieces to the 8 kinging tiles and then verify that the pieces have been made into kings.

testJump() ensure that jump logic is working correctly. If a piece is jumped, it should be removed from the board and the tile it was on should update to reflect this.