Steven Chu

905-094-800

CS 31 Project 7 Report

1. One significant challenge I faced was the issue of the game sometimes entering an infinite loop when the value of the next spot required became 12 and the humanPlay() or computerPlay() functions were called. This is because the next spot required value cannot be incremented beyond 12, and this would result in the condition I set for the loop to always be satisfied, and therefore never end. I solved this issue by implementing a counter that would stop the loop after every possible dice combination is checked multiple times to ensure a valid roll is detected.
2. Using the sample test cases, I was able to test the humanPlay and computerPlay functions that accepted dice parameters.

d1.setValue( 1 );

d2.setValue( 2 );

d3.setValue( 3 );

d4.setValue( 4 );

d5.setValue( 5 );

d6.setValue( 6 );

game.humanPlay( d6, d5, d4 );

human = game.getHuman( );

assert( human.whatSpotIsNeededNext( ) == 1 );

game.computerPlay( d1, d2, d3 );

computer= game.getComputer( );

assert( computer.whatSpotIsNeededNext( ) == 7 );

game.humanPlay( d4, d2, d1 );

human = game.getHuman( );

assert( human.whatSpotIsNeededNext( ) == 8 );

game.computerPlay( d5, d2, d1 );

computer = game.getComputer( );

assert( computer.whatSpotIsNeededNext( ) == 9 );

game.humanPlay( d6, d2, d3 );

human = game.getHuman( );

assert( human.whatSpotIsNeededNext( ) == 10 );

These asserts were used among other starting dice combinations to ensure that these functions worked. To test the board functions, the following asserts were used:

assert( b.getHumanSpot( ) == 0 );

assert( b.getComputerSpot( ) == 0 );

assert( b.isGameOver( ) == false );

assert( b.isHumanWinner( ) == false );

b.setHumanSpot( 3 );

b.setComputerSpot( 6 );

assert( b.getHumanSpot( ) == 3 );

assert( b.getComputerSpot( ) == 6 );

assert( b.isGameOver( ) == false );

assert( b.isHumanWinner( ) == false );

Among others.

Lastly, we can use assert to test the getter and setter functions

assert( !p.hasRolledTen( ) );

assert( !p.hasRolledEleven( ) );

assert( !p.hasRolledTwelve( ) );

assert( p.whatSpotIsNeededNext( ) == 1 );

// now the player has rolled 1... so the spot next needed is 2...

p.rolled( 1 );

assert( p.hasRolledOne( ) );

assert( !p.hasRolledTwo( ) );

assert( p.whatSpotIsNeededNext( ) == 2 );

Among others.

To test the remaining functions in the Centennial class, I found it sufficient to repeatedly run the looping driver code provided to play the game using randomly generated values.