Edmund Zhi

UID: 805328871

Project 7 Report

1. My first error was not realizing a skeleton code was provided, so I wasted several hours trying to make the game for myself. The only other obstacle I encountered was in my logic for BeatThat.determineGameOutcome(). At first, I had written my if statements to check for board.isGameOver, but I was failing test cases. I realized that this was a problem because isGameOver isn’t going to be correct until the function .endTurn runs, which has an updateBoard() part. To fix my problem, rather than checking .isGameOver, I checked whether the human or computer had more rounds won by the time the turns were out.
2. Test Cases:

|  |  |
| --- | --- |
| BeatThat g;  assert( g.turnsLeft( ) == 5 );  assert( !g.isGameOver( ) );  assert( g.determineGameOutcome( ) == cs31::BeatThat::GAMENOTOVER ); | Makes sure that BeatThat.isGameOver() can run independently of Board.isGameOver() |
| Die d1;  d1.setValue( 1 );  Die d2;  d2.setValue( 2 );  Player p;  p.roll( d1, d2 );  assert( p.largestDie( ).getValue( ) == 2 );  assert( p.smallestDie( ).getValue( ) == 1 ); | Makes sure .largestDie() and .smallestdie() in Player.cpp was implemented correctly |
| Player p; assert( p.getRoundsWon( ) == 0 ); p.wonARound(); assert( p.getRoundsWon( ) == 1 ); p.wonARound(); assert( p.getRoundsWon( ) == 2 ); | Makes sure .getRoundsWon() and .wonARound() was implemented correctly |
| Board b; b.setHumanRoundsWon( 3 ); b.setComputerRoundsWon( 2 ); b.markHumanAsWinner(); b.setTurnsLeft(0); assert( b.getHumanRoundsWon() == 3); assert( b.getComputerRoundsWon() == 2 ); assert( b.getTurnsLeft() == 0 ); assert( b.didHumanWin() ); assert( !b.didComputerWin() ); assert( b.isGameOver() ); | Makes sure functions in Board.cpp are working |
| Board b; b.setHumanRoundsWon( 2 ); b.setComputerRoundsWon( 3 ); b.markComputerAsWinner(); b.setTurnsLeft(0); assert( b.getHumanRoundsWon() == 2 ); assert( b.getComputerRoundsWon() == 3 ); assert( b.getTurnsLeft() == 0 ); assert( b.didComputerWin() ); assert( !b.didHumanWin() ); assert( b.isGameOver() ); | Makes sure .markComputerAsWinner is working |
| Board b; b.setHumanRoundsWon( 1 ); b.setComputerRoundsWon( 1 ); b.markTied(); b.setTurnsLeft(0); assert( b.getHumanRoundsWon() == 1 ); assert( b.getComputerRoundsWon() == 1 ); assert( b.getTurnsLeft() == 0 ); assert( !b.didComputerWin() ); assert( !b.didHumanWin() ); assert( b.isGameOver() ); | Makes sure .markTied is working |
| BeatThat g; Die d1; d1.setValue( 1 ); Die d2; d2.setValue( 2 ); Die d3; d3.setValue( 3 ); Die d4; d4.setValue( 4 ); Die d5; d5.setValue( 5 ); g.humanPlay( d1, d2 ); g.computerPlay( d3, d4 ); g.endTurn( ); g.humanPlay( d3, d5 ); g.computerPlay( d3, d4 ); g.endTurn( ); assert( g.turnsLeft( ) == 3 ); assert( !g.isGameOver( ) ); assert( g.determineGameOutcome( ) == cs31::BeatThat::GAMENOTOVER ); assert( g.getHuman().getRoundsWon( ) == 1 ); assert( g.getComputer().getRoundsWon( ) == 1); | Makes sure rounds of play are handled correctly |
| BeatThat g; Die d1; d1.setValue( 1 ); Die d2; d2.setValue( 2 ); Die d3; d3.setValue( 3 ); Die d4; d4.setValue( 4 ); Die d5; d5.setValue( 5 ); Die d6; d6.setValue( 6 ); g.humanPlay( d1, d2 ); g.computerPlay( d3, d4 ); g.endTurn( ); g.humanPlay( d3, d5 ); g.computerPlay( d3, d4 ); g.endTurn( ); g.humanPlay( d1, d2 ); g.computerPlay( d2, d1 ); g.endTurn( ); g.humanPlay( d3, d5 ); g.computerPlay( d5, d3 ); g.endTurn(); g.humanPlay( d1, d6 ); g.computerPlay( d6, d1 ); g.endTurn( ); assert( g.turnsLeft( ) == 0 ); assert( g.isGameOver( ) ); assert( g.determineGameOutcome( ) == cs31::BeatThat::TIEDGAME ); assert( g.getHuman().getRoundsWon( ) == 1 ); assert( g.getComputer().getRoundsWon( ) == 1); | Makes sure declaring winner is done correctly |