Tyler Stovsky

705512370

1. The only notable obstacle that I had to overcome was with the determineGameOutcome function form the BeatThat class. Originally, I tried to check the current game outcome by calling the didHumanWin and didComputerWin functions from the board class, however this led to the game outcome being a tie every time. This happened because those two functions only were updated after a call to isGameOver from the BeatThat class, which only happened after the determineGameOutcome function is already called. To fix this, I used the getRoundsWon function from the Player class in order to check if the number of rounds won for either the human or computer was greater than the other (after checking if 5 rounds have been completed).

**Die Class**

d.roll() // Random toss

d.setValue(0); // Set a value equal to 0

d.setValue(-5); // Set a negative value

d.setValue(7); // Set a value greater than the number of sides

d.setValue(3); // Valid toss

assert(d.getValue() == 3); // Value gets changed when calling a valid setValue

**Player Class**

Player p;

assert(p.getRoundsWon() == 0); // Should be zero at the start of the game

p.wonARound(); // Increments the rounds won counter

assert(p.getRoundsWon() == 1); // Calling wonARound should increase the rounds won

p.roll(); // This should be a random toss

Die d3; d3.setValue(3);

Die d5; d5.setValue(5);

p.roll(d3, d5); // Cheat to set the die values to 3 and 5

assert(p.getDie1().getValue() == 3); // Should return the value of the first die

assert(p.getDie2().getValue() == 5); // Should return the value of the second die

assert(p.largestDie().getValue() == 5); // Check that the largest value is 5

assert(p.smallestDie().getValue() == 3); // Check the the smallest value is 3

p.roll(d3, d3);

assert(p.largestDie().getValue() == 3); // If there is a tie, the largest value and

assert(p.smallestDie().getValue() == 3); // smallest value should be the same

**Board Class**

Board b;

assert(!b.isGameOver(); // At the beginning, the should not be over

assert(!b.didComputerWin()); // At the beginning, the computer has not won

assert(!b.didHumanWin()); // At the beginning, the human as not won

assert(b.getTurnsLeft() == 0); // Turns left should be initialized to zero

assert(b.getHumanRoundsWon() == 0); // Human rounds won should be initialized to zero

assert(b.getComputerRoundsWon() == 0); // Computer rounds won should be initialized to zero

b.setHumanRoundsWon(-5); // Set the rounds won equal to a negative number

b.setHumanRoundsWon(123); // Set the rounds won equal to a number greater than 5

b.setComputerRoundsWon(-9);

b.setComputerRoundsWon(23);

b.setHumanRoundsWon(3); // Set rounds won to a number >= 0 and <= 5

b.setComputerRoundsWon(1);

b.setTurnsLeft(4);

assert(b.getTurnsLeft() == 4);

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

assert(b.getComputerRoundsWon() == 1);

b.markHumanAsWinner();

assert(b.didHumanWin());

assert(!b.didComputerWin());

assert(b.isGameOver()); // If marked as winner, the game is over

b.markComputerAsWinner();

assert(!b.didHumanWin());

assert(b.didComputerWin());

assert(b.isGameOver());

b.markTied();

assert(!b.didHumanWin());

assert(!b.didComputerWin());

assert(b.isGameOver()); // If marked as tied, the game is over

**BeatThat Class**

BeatThat game;

// Game starts with 5 turns left and not over

assert(game.determineGameOutcome() == cs31::BeatThat::GAMENOTOVER);

assert(game.turnsLeft() == 5);

game.humanPlay(); // Human should randomly roll the die

game.computerPlay() // Computer should randomly roll the die

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);

// FULL 5 ROUND GAME WITH DIFFERENT TEST CASES

// Human and computer tied

game.humanPlay(d4, d6);

game.computerPlay(d6, d4);

game.endTurn();

assert(game.determineGameOutcome() == cs31::BeatThat::GAMENOTOVER);

assert(game.turnsLeft() == 4);

assert(game.getHuman().getRoundsWon() == 0);

assert(game.getComputer().getRoundsWon() == 0);

assert(!game.isGameOver());

// Human wins based on the first dice

game.humanPlay(d2, d6);

game.computerPlay(d5, d4);

game.endTurn();

assert(game.determineGameOutcome() == cs31::BeatThat::GAMENOTOVER);

assert(game.turnsLeft() == 3);

assert(game.getHuman().getRoundsWon() == 1);

assert(game.getComputer().getRoundsWon() == 0);

assert(!game.isGameOver());

// Computer wins based on the first dice

game.humanPlay(d2, d3);

game.computerPlay(d5, d4);

game.endTurn();

assert(game.determineGameOutcome() == cs31::BeatThat::GAMENOTOVER);

assert(game.turnsLeft() == 2);

assert(game.getHuman().getRoundsWon() == 1);

assert(game.getComputer().getRoundsWon() == 1);

assert(!game.isGameOver());

// Human wins based on the second dice

game.humanPlay(d5, d3);

game.computerPlay(d5, d1);

game.endTurn();

assert(game.determineGameOutcome() == cs31::BeatThat::GAMENOTOVER);

assert(game.turnsLeft() == 1);

assert(game.getHuman().getRoundsWon() == 2);

assert(game.getComputer().getRoundsWon() == 1);

assert(!game.isGameOver());

// Computer wins based on the second dice

game.humanPlay(d5, d2);

game.computerPlay(d4, d5);

game.endTurn();

assert(game.isGameOver());

assert(game.turnsLeft() == 0);

assert(game.getHuman().getRoundsWon() == 2);

assert(game.getComputer().getRoundsWon() == 2);

assert(game.determineGameOutcome() == cs31::BeatThat::TIEDGAME);

// NEW GAME

// Human wins

game.humanPlay(d2, d6);

game.computerPlay(d5, d4);

for (int i = 0; i < 5; i++) {

game.endTurn();

}

assert(game.isGameOver());

assert(game.turnsLeft() == 0);

assert(game.getHuman().getRoundsWon() == 5);

assert(game.getComputer().getRoundsWon() == 0);

assert(game.determineGameOutcome() == cs31::BeatThat::HUMANWONGAME);

// NEW GAME

// Computer wins

game.humanPlay(d1, d3);

game.computerPlay(d3, d4);

for (int i = 0; i < 5; i++) {

game.endTurn();

}

assert(game.isGameOver());

assert(game.turnsLeft() == 0);

assert(game.getHuman().getRoundsWon() == 0);

assert(game.getComputer().getRoundsWon() == 5);

assert(game.determineGameOutcome() == cs31::BeatThat::COMPUTERWONGAME);