Kevin Wang

UID: 205209507

Project 7 Report

Notable obstacles:

* Initially I had trouble finding out what the problem with my code was when I tried to test the asserts for whatSpotIsNeededNext within the centennial game because through debugging, it would always return the default values and not the ones that were supposed to change after humanPlay() or computerPlay(). I worked around both the rolled() and whatSpoIsNeededNext() operations within the player class for awhile and still couldn’t figure out the issue until I checked my centennial.cpp file and realized that I missed a crucial step of assigning the mHuman and mComputer to the local variable of those types that were changed. After making sure the member values of the game were updated, I was able to have the changes done to the player and computer objects saved to the game member values.
* Because the game requires you to roll 3 dice values but there are actually 7 possible outcomes from the 3 die values, I had trouble initially coming up with a way to get all 7 values to be checked within the game. I settled for a solution that created an array with all 7 combinations and created a loop that have a pointer variable to each spot in the array to transverse through all the possible outcomes.

Test Data

The player.cpp file was tested using the given asserts:

// Player test code

Player p, human, computer;

// in the beginning of time, nothing has been rolled yet and the spot needed is 1...

assert( !p.hasRolledOne( ) );

assert( !p.hasRolledTwo( ) );

assert( !p.hasRolledThree( ) );

assert( !p.hasRolledFour( ) );

assert( !p.hasRolledFive( ) );

assert( !p.hasRolledSix( ) );

assert( !p.hasRolledSeven( ) );

assert( !p.hasRolledEight( ) );

assert( !p.hasRolledNine( ) );

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 );

// only rolls from 1-12 are relevant...

p.rolled( 100 );

assert( p.hasRolledOne( ) );

assert( !p.hasRolledTwo( ) );

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

// rolls must be sequential for things to count...

p.rolled( 3 );

assert( p.hasRolledOne( ) );

assert( !p.hasRolledTwo( ) );

assert( !p.hasRolledThree( ) );

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

// work the Player via Dies

d1.setValue( 6 );

d2.setValue( 5 );

d3.setValue( 4 );

p.roll( d1, d2, d3 );

assert( p.whatWasRolled() == "Die1: 6 Die2: 5 Die3: 4\n" );

The boad.cpp was tested using the given asserts:

// Board test code

Board b;

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 );

b.setHumanSpot( 12 );

assert( b.getHumanSpot( ) == 12 );

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

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

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

b.setGameOver( true );

b.markHumanAsWinner( );

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

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

The Centennial Test Code

The centennial.cpp cheating functions were tested using the given asserts:

Centennial game;

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

assert( game.determineGameOutcome( ) == Centennial::GAMENOTOVER );

human = game.getHuman( );

computer = game.getComputer( );

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

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

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 );

game.computerPlay( d1, d2, d3 );

computer = game.getComputer( );

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

game.humanPlay( d4, d5, d6 );

human = game.getHuman( );

assert( human.whatSpotIsNeededNext( ) == 12 );

game.computerPlay( d3, d2, d1 );

computer = game.getComputer( );

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

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

assert( game.determineGameOutcome( ) == Centennial::GAMENOTOVER );

game.humanPlay( d2, d4, d6 );

assert( game.isGameOver( ) == true );

assert( game.determineGameOutcome( ) == Centennial::HUMANWONGAME );

additional asserts were added to check that the board was updated correctly within the cheating function

game.computerPlay(d1, d2, d3);

computer = game.getComputer();

b = game.getBoard();

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

game.humanPlay(d4, d2, d1);

human = game.getHuman();

b = game.getBoard();

assert(b.getHumanSpot() == 7);

humanPlay() and computerPlayer() where they were to randomly rolled was texted using the Skelton code provided within the project

using namespace cs31;

using namespace std;

clearScreen();

Centennial game;

std::string action, message = "(r)oll (q)uit: ";

std::cout << message;

// for now...

Die d1;

Die d2;

Die d3;

do

{

getline(cin, action);

while (action.size() == 0)

{

getline(cin, action); // no blank entries allowed....

}

switch (action[0])

{

default: // if bad move, nobody moves

cout << '\a' << endl; // beep

continue;

case 'Q':

case 'q':

return 0;

case 'r':

case 'R':

game.humanPlay();

cout << game.display("") << endl;

if (!game.isGameOver())

{

game.computerPlay();

cout << game.display(message) << endl;

}

break;

}

} while (!game.isGameOver());

return(0);