David Lane

004638553

Project 7 report

1. There were two points in this project where I had some trouble. The first was getting the game to show an X when a round was won. Initially, in the determineRoundOutcome Bunco class member function I was incrementing the score or changing the winner manually instead of calling the markHumanAsWinner function which was what I needed to make the X appear because that would then call a certain function in BoardRow which would change the board rows display.

The second problem I had was that initially for the determineGameOutcome Bunco class member function I was checking to see if the round was greater than 7 to determine if the game was over but when I tried some test cases I realized that they were not necessarily incrementing the round again once the 6th round was complete, so to use more solid logic I decided to check if the players total round wins was equal to six to check if the game was over instead.

1. To test my solution to this project, I mostly just played the game a bunch of times, ensuring several things:
2. The first player to roll the round number won that round
3. In case of a tie, game prompts a reroll
4. Once a winner is determined for the round, adds an x to the correct side and moves to next round
5. Make sure only 6 rounds occurred
6. Check each possible endgame scenario (all results for the determineGameOutcome Bunco class member function) and make sure they are all working properly

Then I supplemented my play trials with Howards Test code to make sure I had adequate compatibility for the forced rolls and cassertions.

#include <iostream>

#include <string>

#include <cassert>

#include "Die.h"

#include "Player.h"

#include "Bunco.h"

int main()

{

using namespace std;

using namespace cs31;

    // test code

    Die d;

    for (int i = 1; i <= 100; i++)

    {

        d.roll();

        int value = d.getValue();

        assert( value >=1 && value <= 6 );

    }

       Player p;

    assert( p.getScore() == 0 );

    p.setRound( 1 );

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6 );

    assert( p.getScore() == 0 );

    assert( p.roll( 5 ) == 5 );

    assert( p.getScore() == 0 );

    assert( p.roll( 1 ) == 1);

    assert( p.getScore() == 1 );

    p.setRound( 6 );

    assert( p.getScore() == 0 );

    assert( p.roll( 6 ) == 6 );

    assert( p.getScore() == 1 );

    Bunco b;

    b.setRound( 1 );

    assert( b.determineRoundOutcome() == Bunco::NOTDECIDED );

    b.computerPlay( 5 );

    b.humanPlay( 5 );

    assert( b.determineRoundOutcome() == Bunco::NOTDECIDED );

    b.computerPlay( 1 );

    b.humanPlay( 1 );

    assert( b.determineRoundOutcome() == Bunco::NOTDECIDED );

    b.computerPlay( 1 );

    b.humanPlay( 2 );

    assert( b.determineRoundOutcome() == Bunco::COMPUTERWON );

     cout << "all tests passed!" << endl;

    return 0;

}