Brendan Jang
CS 162 Lab 3
1/24/2019

1. Brief description
    a. The program contains 4 classes
        i. Game
        ii. Die
        iii. LoadedDie
        iv. Menu
    b. Die holds information of a die which is the number of its sides
    c. Loaded Die is derived from Die, and it sometimes gets a higher rolling result than Die does. We did it by overloading a function of Die in LoadedDie named getRandom.
    d. Game fulfills all tasks which are stated in the assignment
    e. Menu is used to reduce the amount of repeated code which is used to validate the input from the user
2. Changes made during implementation
    a. I believe that the original design was good, so I did not have to cope with any design problems while implementing the game.
3. Problems encountered during implementation
    a. There weren't any major problems encountered during implementation.
    b. The main issues that I was encountering was that I was mixing up variable names due to typos and small syntax errors that were easily fixed by following the error messages on putty.
4. Test table

Test case	Input value	Driver function	Expected Outcome	Observed Outcome
Invalid input	r	Menu::getValue(string* messages, int size, int min_val, int max_val)	Loop back to the question prompting the user for input	Loop back to the question prompting the user for input
Invalid input	min_val - 1	Menu::getValue(string* messages, int size, int min_val, int max_val)	Loop back to the question prompting the user for input	Loop back to the question prompting the user for input
Invalid input	max+val + 1	Menu::getValue(string* messages, int size, int min_val, int max_val)	Loop back to the question prompting the user for input	Loop back to the question prompting the user for input
Stop game	2	void Game::start()	Game ends	Game ends
Die vs Die	6 12 1 10 1	void Game::play()	Cannot predict the output	---------- Rolling result Score result Round Player_1 Player_2 Player_1 Player_2 1 8 7 1 0 2 10 6 1 0 3 6 6 0 0

				4 11 3 1 0 5 10 2 1 0 6 3 8 0 1 ---------- Final score: Player_1: 4 Player_2: 1 The final winner is: Player_1
LoadedDie vs LoadedDie	6 6 2 6 2	void Game::play()	Cannot predict the output	---------- Rolling result Score result Round Player_1 Player_2 Player_1 Player_2 1 2 5 0 1 2 4 2 1 0 3 6 2 1 0 4 5 1 1 0 5 4 2 1 0 6 3 2 1 0 ---------- Final score: Player_1: 5 Player_2: 1 The final winner is: Player_1
Die vs LoadedDie	6 15 1 15 2	void Game::play()	Cannot predict the output	---------- Rolling result Score result Round Player_1 Player_2 Player_1 Player_2 1 14 2 1 0 2 13 11 1 0 3 9 11 0 1 4 2 13 0 1 5 10 2 1 0 6 3 8 0 1 ---------- Final score: Player_1: 3 Player_2: 3 The final winner is: None