Lab 3

Submit Assignment

Due Sunday by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip

War Game with Dice Design

Goals

- Identify requirements for a program
- Develop an initial test plan to test program requirements
- Implement the game according to your design
- Program using inheritance

For this lab, we will be designing a simplified version of **War** using **dice**.

War Game Requirement

Setting

This is a 2-player game. It is played through dice.

Rule for scoring

The player who rolls higher number gets one point. If both players roll the same number, it is considered a draw and no one gets a point.

Dice Specification

There are two kinds of dice:

- normal die, represented by **Die class.**
- loaded die, represented by the **loadedDie class**.

Classes

Die class

Die class has a member variable of an **integer N** that represents the **number of sides on the individual die.**

Die class also has a **member function** that returns a random integer between 1 and N as the result of rolling the die once.

LoadedDie class

LoadedDie class **inherits** the behavior and elements of Die class.

However, for the die rolling function, the number it returns is **biased** such that **the average output of rolling it for several times is higher** than that of a Die object with the same number of sides.

You can determine how you want to make the die loaded, as long as it fulfills the requirement above.

Note: A loaded Die can **never roll a number that is higher than the number of sides it has**. So, if a loaded die has 6 sides, it cannot roll a 10.

Game class:

This class implements the dice-rolling war game.

First, create a **menu** that first asks the user to select from the following 2 choices:

- Play game
- Exit game

If the user selects “play game”, ask the following questions at the start of the game:

- How many rounds will be played
- The type of die for each player
- The number of sides for dice of both players

If the user selects “Exit game”, exit the game.

Note: The two dice can have different numbers of sides.

Note: You are allowed to add more choices for your menu as long as it makes sense.

Design the menu so it makes it easier for the user to read, and input selection.

After getting all the information from the user, the game then creates the necessary Die/LoadedDie objects, and play the game.

During the game, the game should output the **detailed result** of each round, including the **side and type of die used for each player, the number each player rolls, and the score result.** Afterwards, **display the final score count** after results of all rounds are printed, and the **final winner** of the game.

Note: You are allowed to have other classes, as long as you have the classes specified above, and the behavior satisfies the above requirements.

Input Validation

In lab 1, we briefly talked about the requirement of input validation. For this lab, we will be implementing input validation for all inputs.

Input validation is the testing for input that is supplied from outside source. (including human input)

Consider the following scenario: if your game request for an input of integer and the user instead input a character of “t”, it would cause the program to crash. But with an input validation, the error is caught, and the input is requested again, until the user input a correct type of data.

The requirement of input validation is to make sure the program

- **does not crash from the undesired input**
- **request for input repeatedly until the correct data is inputted.**

A good way of planning input validation is to first think about what kind of input is desired for each input in a program.

If you would like to add more feature to your input validation, you are free to do so.

If you have any question, there will be a discussion section on Piazza.

Reflection Document

- Briefly describe your original design.
- Talk about the changes made during implementation of the game.
- Talk about the problems encountered during implementation, and how you solved them.
- Include your test table that includes test plan, and the test results.

Memory Leaks

Starting from this lab, TAs will start to **deduct points** for memory leaks. Make sure to check memory leaks and test your program thoroughly on the flip server.

What you need to submit:

- All the program files including header and source files (.cpp/.hpp)
- Your makefile
- Your reflection pdf file

Important: Put all the files in a single .zip file and submit it on Canvas.

Grading

- Programming style – 10% (having memory leaks will lose points here)
- Correctly implement and use the Game class – 20%
- Correctly implement and use the Die class – 20%
- Correctly implement and use the LoadedDie class – 20%
- Include your menu correctly – 10%
- Include your input validation function for user input correctly – 10%

- Your reflection document – 10% (You need to include test results using different combinations of Die and LoadedDie for each player and different numbers of sides for each player in the test plan table in the reflection document)

Lab 3

Criteria	Ratings	Pts
Programming Style Commenting, meaningful function/variable names, memory leaks - no memory leaks		10.0 pts
Game Class Correctly implement and use the Game class - play game / exit game menu - how many rounds, type of die for each player - number of sides of dice for each player - can be different - output detailed results for each round during game (side and type of player die, the number each player rolls, and the score result). - output final score and winner at the end of game.		20.0 pts
Die Class Correctly implement and use the Die class. - Has N - number of sides integer - Has roll function that returns random int between 1 and N		20.0 pts
LoadedDie Class Correctly implement and use the LoadedDie class. - Inherits Die class - Overloads rolling function with biased version		20.0 pts
Menu Include your menu correctly		10.0 pts
Validation Include your input validation function for user input correctly - doesn't crash on float input - doesn't crash on char / string input		10.0 pts
Reflection Document You need to include test results using different combinations of Die and LoadedDie for each player and different numbers of sides for each player in the test plan table in the reflection document - describe original design - describe changes from original design - describe problems encountered and solutions - test table / test plan, test results		10.0 pts
		Total Points: 100.0