Lab 3
Yucen Cao

Design description:

In this project, it is a 2-player game. It is played through dice. 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.There are two kinds of dice:

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

For the Die class, it has N1 sides, returns a random integer between 1 and N as the result of rolling the die once.
For the LoadedDie, it has N2 sides, returns an integer between 1 and N. 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.

Project Structure
It has main, die_class, loadedDie_class, game_class, menu, and input_validation. They can split to 3 important parts, 1.user input and validation. 2. Die and LoadedDie return number. 3. Game logic and print

My design

- In the menu, when user input 0, then it will exit the game.
- I use an int array to store the user's input. The size is 5.
  input[0]: the number of the rounds
  input[1]: the type of die for player A (Die is 0 /LoadedDie is 1)
  input[2]: the type of die for player B (Die is 0 /LoadedDie is 1)
  input[3]: the number of sides for dice for player A
  input[4]: the number of sides for dice for player B
- The logic of LoadedDie, because 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. When the sides is N, I first get the random integer between 1 and N, after that it will have 50% chance to add 1 by using return random integer between 0 and 1. If the return num is N+1, then return N.
- For every round, the output will show like in this way

```
NO.1 rounds: Winner is B.
The number rolled by A: 4  Side of A: 6  Type of A: LoadedDie
The number rolled by B: 7  Side of B: 7  Type of B: Die
The score result, A:B = 0:1
```

Test table:

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Input float number	1.5	bool validation_int(string s) ;	Show enter wrong number, please enter again	Show enter wrong number, please enter again
Input character	A or bb or #	bool validation_int(string s) ;	Show enter wrong number, please enter again	Show enter wrong number, please enter again
Input negative number	-12	bool validation_int(string s) ;	Show enter wrong number, please enter again	Show enter wrong number, please enter again
Input 0 for steps and sides	0	bool validation_int(string s) ; int validation_positive_int(string temp) ；	Show enter wrong number, please enter again （for steps and sides it can not be 0)	Show enter wrong number, please enter again
Input not only 0 or 1 for the choose parts.	2	bool validation_int(string s) ; int validation_one_zero(string temp);	Show enter wrong number, please enter again （when user choose something, it has to be 0/1 according to the design.	Show enter wrong number, please enter again
Input 0 for start the game	0	bool validation_int(string s) ; int validation_one_zero(string temp);	Exit the game	Exit the game
input 9	9	void		

rounds;
Input die for A and B;
side of 6 for both
	0
	0
	6
	6
		Game::run_game(int *input);	Show round for each player, then show the final result. A/B will win	Show round for each player, then show the final result. A win

Input 200 rounds; Input die for A, Loadeddie for B. Side 8 for A, side 18 for B	200 0 1 8 18	void Game::run_game(int *input);	Show round for each player, then show the final result. B has more chance to win, but A still have chance to win	Show round for each player, then show the final result. B win.
Input 200 rounds; Input die for A, Loadeddie for B. Side 18 for A, side 8 for B	200 0 1 18 8	void Game::run_game(int *input);	Show round for each player, then show the final result. A/B will win	Show round for each player, then show the final result. B win.

Input 45000 rounds;
Input LoadedDie for A, Die for B;
Side 18 for both of A and B
	45000
	1
	0
	8
	8
		void Game::run_game(int *input);	Show round for each player, then show the final result. A has more chance to win	Show round for each player, then show the final result. A win.

Input 45000 rounds;
Input LoadedDie for B, Die for A;
Side 18 for both of A and B
	45000
	0
	1
	9
	9
		void Game::run_game(int *input);	Show round for each player, then show the final result. B has more chance to win	Show round for each player, then show the final result. B win.

Input 45000 rounds;
Input LoadedDie both A and B;
Side 18 for
	45000
	1
	1
	18
	108
		void Game::run_game(int *input);	Show round for each player, then show the final result. B has more chance to win	Show round for each player, then show the final result. B win.

A, and side 108 for B				

Input 1000 rounds;
Input Die both A and B;
Side 72 for A, and side 7 for B
	1000
	0
	0
	72
	7
		void Game::run_game(int *input);	Show round for each player, then show the final result. A has more chance to win	Show round for each player, then show the final result. A win.

Reflection:

I first have the different logic for the LoadedDie, that first get random number from 1 and N, then add 1. But I realise that, in this way, number 1 will never show. So I change my logic, the logic is first get random number from 1 and N, then it will have 50% change add 1. In this way, 1 will have the chance to show.

After set down the LoadedDie logic, I write two class. One is for Die, and LoadedDie is the derived class for Die.

I have trouble with user choose the Die and LoadedDie. Before the enter, we don't know what is the type. I first use if statement, and put Die playerA / LoadedDie playerB in the statement to define a new Die or LoadedDie. But it doesn't work. So I use the point, and virtual function.