Pranav Thombare 180859562

Queen Mary University of London

ECS769P-Advanced Object Oriented Programming

Instructions to run the code

```
$g++ -o pontoon *.cpp —std==14
$./pontoon
```

Design

The program for "Pontoon" has 5 source files and their respective header files excluding the main program. The rules for "Pontoon" has been explained in the spec sheet and the program aims to replicate all the scenarios mentioned in the sheet.

As mentioned above, there are 5 source files along with their header files. They are

- Card
- Deck
- Player
- Dealer
- Game

Implementation

The implementation of this program is divided into five parts. Each part has its own significance and its own logic which when put together works according to the spec sheet.

Card

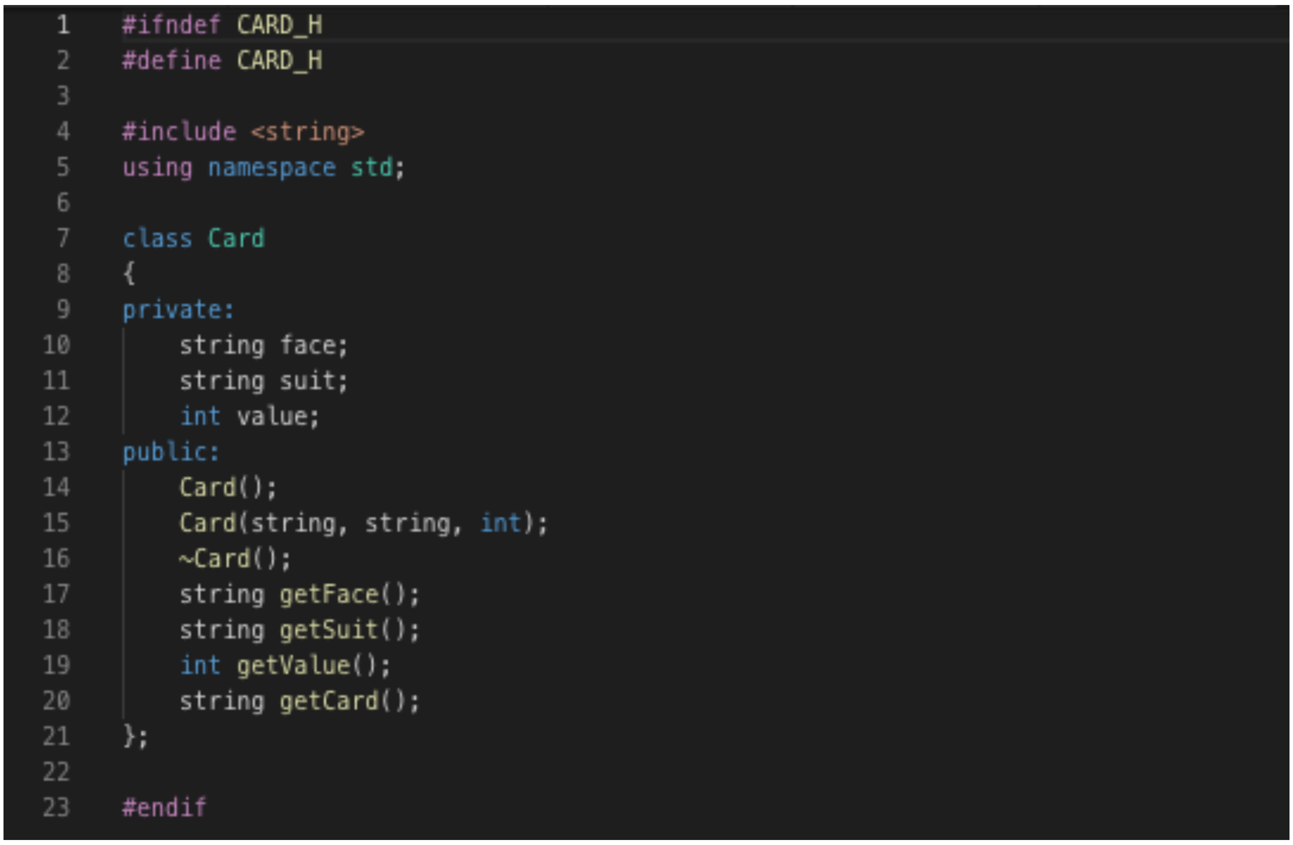
This file is meant to deal with the declaration of cards. The implementation of the cards was tricky as there are many representations of a card deck. One can use stacks to show as a deck of cards. But in this program I chose to use two string variables namely "face" and "suit" to determine the face and suit of any card.

The screenshot above is the class for card. In that, we have three member values which represent the face, suit and the value of a card.

Then there are two constructors, one of them is a default constructor and the other one is a parameterised constructor which helps assign some values to the card.

Deck

The core logic of this program was to come up with a solution to shuffle the cards in such a way that no two cards are repeated while dealing the cards to the player and the dealer(in this case the computer).



```cpp
#ifndef CARD_H
#define CARD_H

#include <string>
using namespace std;

class Card
{
private:
    string face;
    string suit;
    int value;
public:
    Card();
    Card(string, string, int);
    ~Card();
    string getFace();
    string getSuit();
    int getValue();
    string getCard();
};

#endif
```

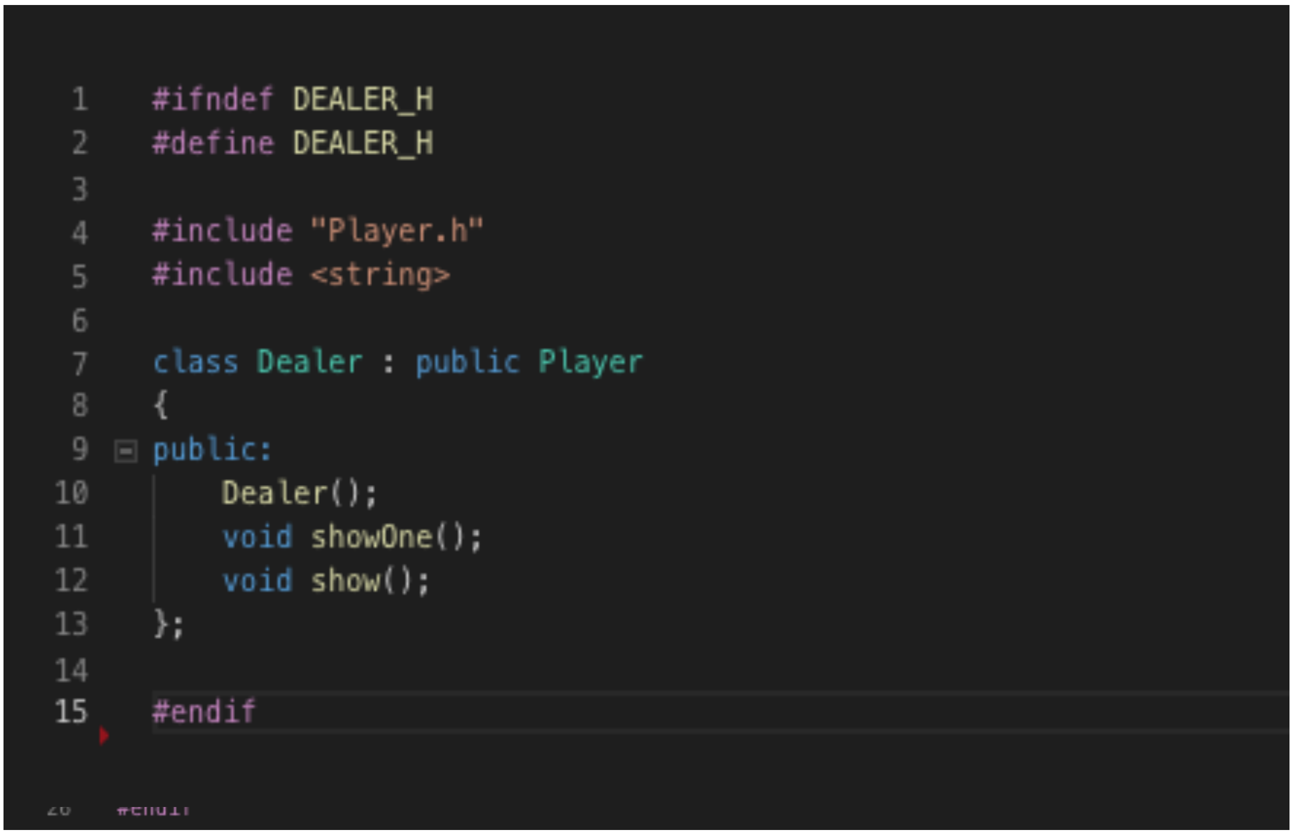
There were many approaches to shuffle a deck. One of the ways to do that was to randomise and form a combination of suite and face. But this resulted in duplication of cards in the same deck. Another way I tried was to assign values to all 52 cards in a deck to a unique string .An example would be to name the ***Ace of Spades*** as "**1S**" and ***eight of hearts*** as "**8H**" . But to handle each value and perform calculations turned out to be a big work later on.

Then I came up with a way to initialise a deck with all values as an array of cards and then perform a random swap for all positions of the deck. This resulted in a random deck being created which was the main task of this program.

Player
This class represents a player in the game. Following is the screenshot of the class with its member functions and variables.
The main trick here was to assign a card from the deck to the player. For that I created a vector of cards which represented the hand of a player with a maximum number of cards he/she can hold being equal to five.

The member functions were used to create a hand, to add a card in an existing hand, to remove the cards in case of a new game and a function to retrieve the value of a card. The class also takes in the name of the Player so that it can use it to store the name in the log file which will be generated in the end of the program.

Dealer
Dealer being one of the players in real life is the reason why dealer inherits the class ***Player***.

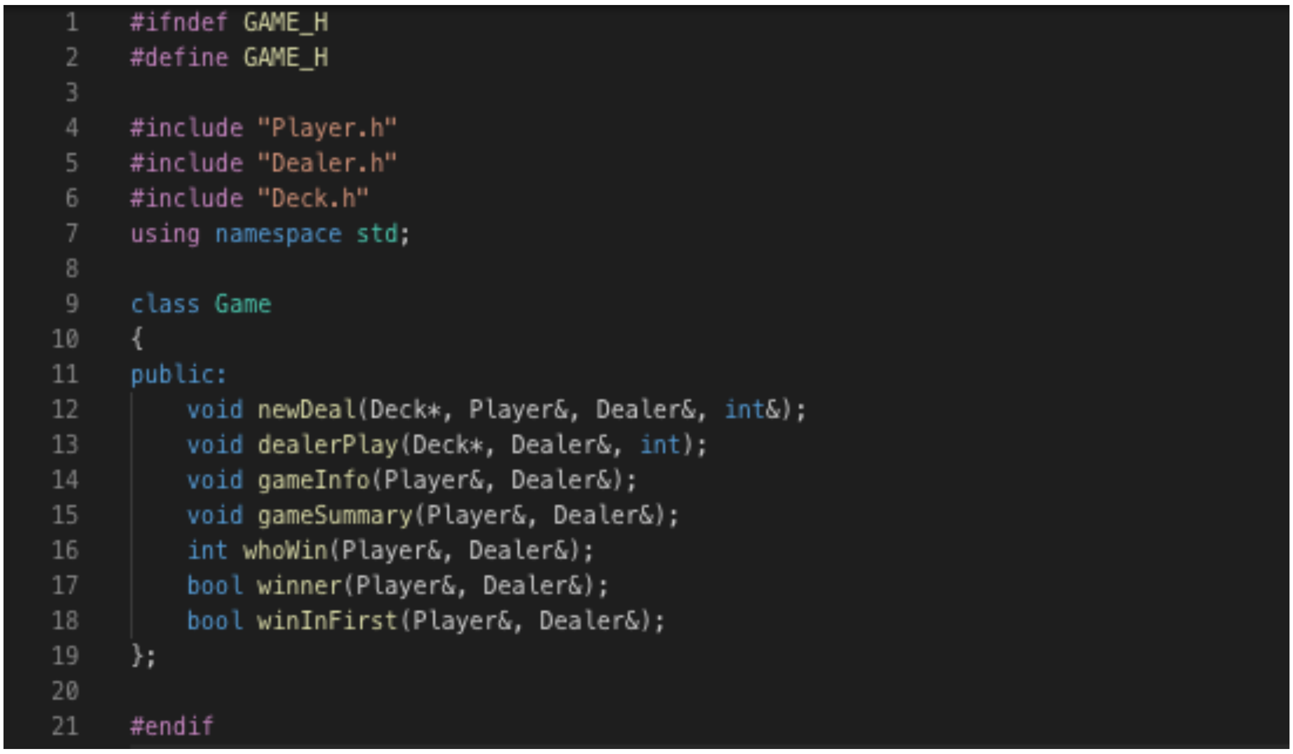


```cpp
#ifndef DEALER_H
#define DEALER_H

#include "Player.h"
#include <string>

class Dealer : public Player
{
public:
    Dealer();
    void showOne();
    void show();
};

#endif
```

The class ***Dealer*** has no member variables but it has 2 member functions and one constructor. The functions ***void showOne()*** and ***void show()*** are similar but the only difference being the former showing only 1 card and the latter showing all the cards for the dealer.

Game

The main logic for how the winner is decided is all written in the ***Game.cpp*** and ***Game.h*** file.

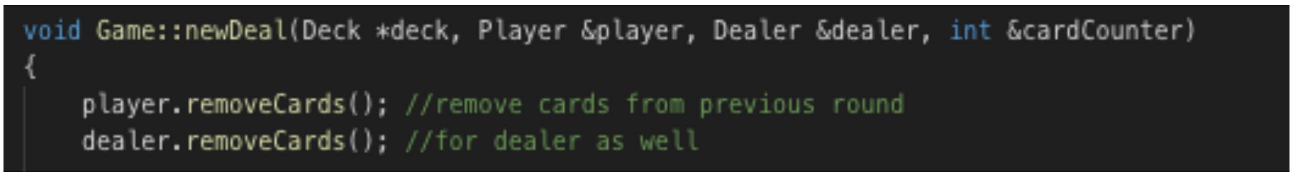


```cpp
#ifndef GAME_H
#define GAME_H

#include "Player.h"
#include "Dealer.h"
#include "Deck.h"
using namespace std;

class Game
{
public:
    void newDeal(Deck*, Player&, Dealer&, int&);
    void dealerPlay(Deck*, Dealer&, int);
    void gameInfo(Player&, Dealer&);
    void gameSummary(Player&, Dealer&);
    int whoWin(Player&, Dealer&);
    bool winner(Player&, Dealer&);
    bool winInFirst(Player&, Dealer&);
};

#endif
```

This class has a lot of member functions. Each of them are discussed below

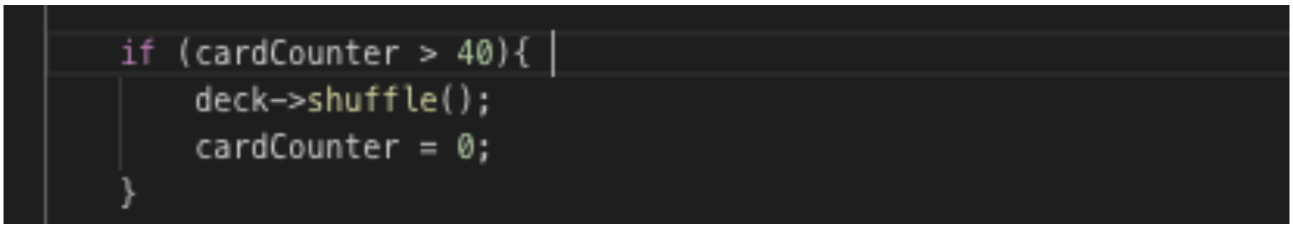
- ***newDeal()***

This method is invoked whenever a new deal is needed.



```cpp
void Game::newDeal(Deck *deck, Player &player, Dealer &dealer, int &cardCounter)
{
    player.removeCards(); //remove cards from previous round
    dealer.removeCards(); //for dealer as well
```

This also removes any previous cards being handed to the player and the dealer. After the cards have been removed, the cards are then distributed one by one starting from player.



```
if (cardCounter > 40){
    deck->shuffle();
    cardCounter = 0;
}
```

A card counter is kept so that whenever more than 40 cards have been dealt, the function to shuffle the cards will be invoked again.

- ***dealerPlay()***

This function is used to deal cards to the dealer.

- ***winner()***

This function is used to determine the winner between the player and the dealer by checking all the possible scenarios (who's score more or less than 21 and who should win).

- ***gameInfo()***

This function shows the cards dealt to the user and the combined values of the card in hand and same for the dealer, except that the dealer's second card is not visible to the player.

- ***gameSummary()***

This function is the same as ***gameInfo()*** except that it unveils all the cards of the dealer.

- ***whoWin()***

This function is an integer based function which returns specific values whenever someone wins based on the condition of the score.

- ***winInFirst()***

This function checks if anyone wins in the first deal of cards.

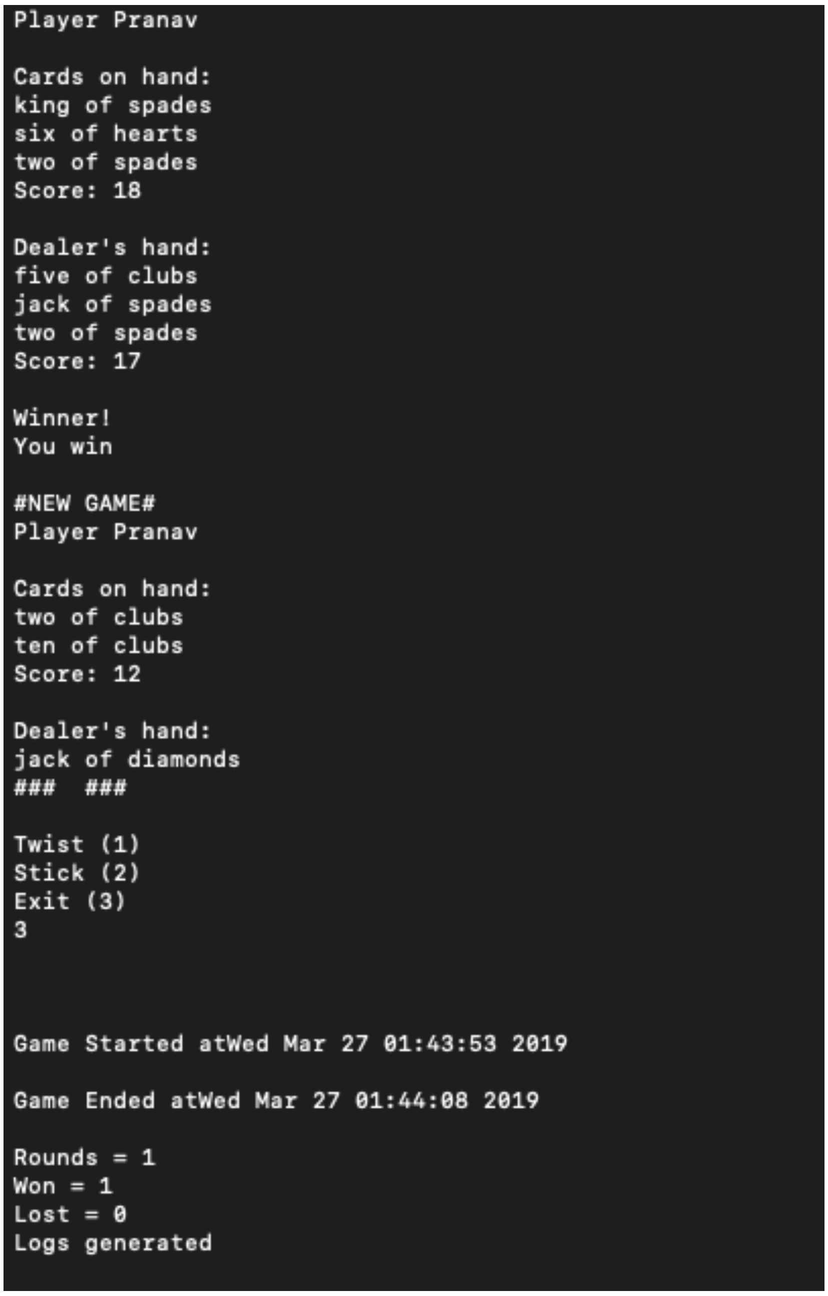
Reusability

The great thing about this program is that the code for the cards and the deck can be used for different games other than Pontoon.

Future Scope

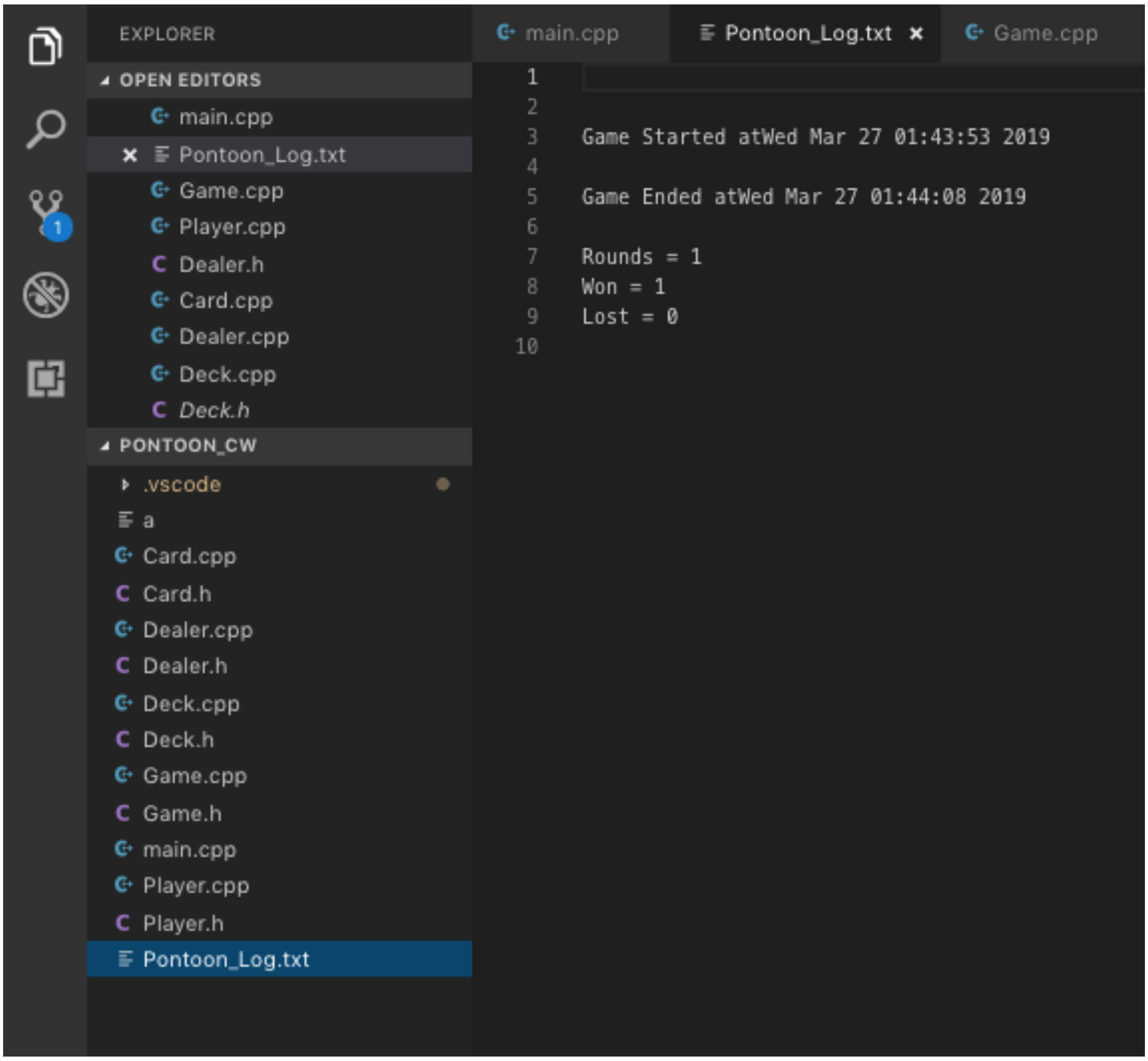
This program can be further modified to accommodate the option to bet and show the winnings after each round. The feature for more decks can also be implemented to imitate the game for real.

Output
The output generated is as follows



```
Player Pranav

Cards on hand:
king of spades
six of hearts
two of spades
Score: 18

Dealer's hand:
five of clubs
jack of spades
two of spades
Score: 17

Winner!
You win

#NEW GAME#
Player Pranav

Cards on hand:
two of clubs
ten of clubs
Score: 12

Dealer's hand:
jack of diamonds
###  ###

Twist (1)
Stick (2)
Exit (3)
3



Game Started atWed Mar 27 01:43:53 2019

Game Ended atWed Mar 27 01:44:08 2019

Rounds = 1
Won = 1
Lost = 0
Logs generated
```

The logs generated were as follow



```


Game Started atWed Mar 27 01:43:53 2019

Game Ended atWed Mar 27 01:44:08 2019

Rounds = 1
Won = 1
Lost = 0

```