資料結構

Data Structure

Lab 04

姓名：曾致嘉

學號：113AB0014

Lab04-Q1

Exercise: Shuffling and Dealing Cards (Use Stack)

1) Shuffle a standard 52-card deck (excluding jokers).

2) Push the shuffled cards into a stack.

3) Draw cards from the stack and display the results.

Code

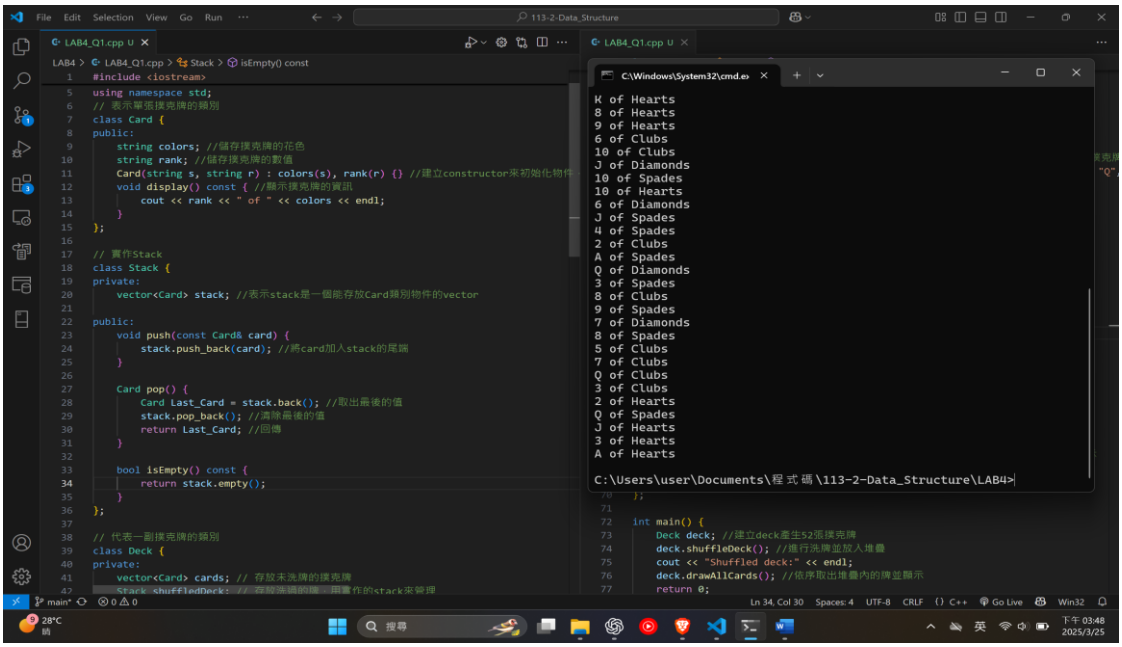
```
#include <iostream>
#include <vector>
#include <algorithm>
#include <random>
#include <ctime>
using namespace std;
// 表示單張撲克牌的類別
class Card {
public:
    string colors; //儲存撲克牌的花色
    string rank; //儲存撲克牌的數值
    Card(string s, string r) : colors(s), rank(r) {} //建立 constructor 來初始化物件，當 Card 物件建立時，它會自動執行這個函式，並把 s 和 r 的值存入 colors 和 rank
    void display() const { //顯示撲克牌的資訊
        cout << rank << " of " << colors << endl;
    }
};

// 實作 Stack
class Stack {
private:
    vector<Card> stack; //表示 stack 是一個能存放 Card 類別物件的 vector

public:
    void push(const Card& card) {
        stack.push_back(card); //將 card 加入 stack 的尾端
    }

    Card pop() {
        Card Last_Card = stack.back(); //取出最後的值
```

```
            stack.pop_back(); //清除最後的值
            return Last_Card; //回傳
        }

        bool isEmpty() const {
            return stack.empty();
        }
};

// 代表一副撲克牌的類別
class Deck {
private:
        vector<Card> cards; // 存放未洗牌的撲克牌
        Stack shuffledDeck; // 存放洗過的牌，用實作的 stack 來管理
public:
        Deck() { //建立 constructor 來初始化物件
            string colors[] = { "Hearts", "Diamonds", "Clubs", "Spades" }; //儲存撲克
牌的花色
            string ranks[] = { "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K",
"A" }; //儲存撲克牌的數值

            //利用迴圈將 52 張牌加入 cards 這個 vector 裡面
            for (int i = 0; i < 4; i++) { // 四種花色
                for (int j = 0; j < 13; j++) { // 13 種點數
                    cards.push_back(Card(colors[i], ranks[j]));
                }
            }
        }

        //洗牌(Hint:使用 shuffle 函數)
        void shuffleDeck() {
            random_shuffle(cards.begin(), cards.end()); //將 cards 裡面的牌打亂
            for (int i = 0; i < 52; i++) {
                shuffledDeck.push(cards[i]); //將打亂的牌加入 shuffledDeck 這個
stack
            }
        }
```

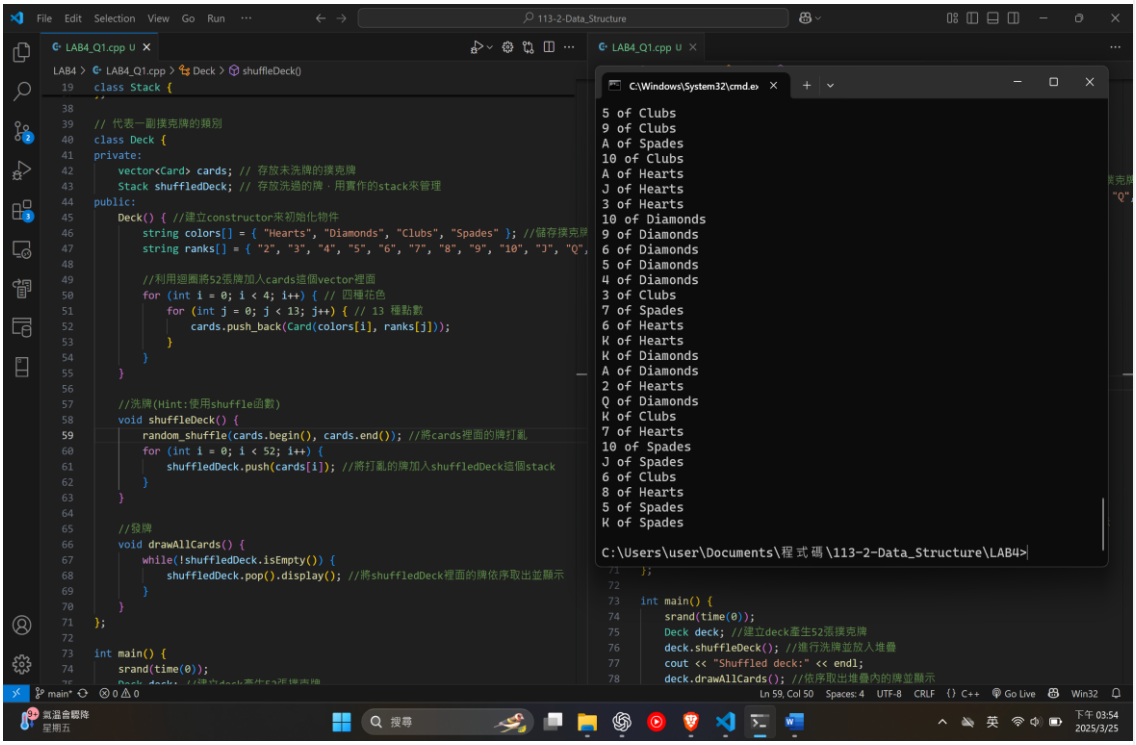
```
    //發牌
    void drawAllCards() {
        while(!shuffledDeck.isEmpty()) {
            shuffledDeck.pop().display(); //將 shuffledDeck 裡面的牌依序取出並顯示
        }
    }
};

int main() {
    srand(time(0));//建立隨機種子
    Deck deck; //建立 deck 產生 52 張撲克牌
    deck.shuffleDeck(); //進行洗牌並放入堆疊
    cout << "Shuffled deck:" << endl;
    deck.drawAllCards(); //依序取出堆疊內的牌並顯示
    return 0;
}
```

Discussion Section

First time



Second time



Third time