Project 1

Twenty One

CIS-17C
Name: Angel Zambrano
Date:4/29/2020

Game Rules

My version of twenty one is a two player game. In each game, the users are given two randomized cards to start with. Afterwards, the user can make the decision of hitting the dealer or staying with the cards they are given. Whoever, reaches or gets closer to 21 wins the game. Note that J, Q and R are all worth ten and A is worth 1 or 11 depending on the user. If the program notices the user has an Ace, the user is asked if he will like to count it as 1 or 11. At the end of the game, the user is asked if he or she will like to play again.

Summary

For this project I chose the card game Twenty One. The reason why I chose this game is because I am very familiar with it and knew that I would not have trouble understanding the game. I had other games planned like Uno Or Set. However, these games require a more visual aspect and I felt that just printing the cards would not be the same.

To finish this program, I took about four days to finish it. The hardest part about this was actually organizing how each class would function. After I had organized that, the rest of my time went onto understanding how to use non-mutating algorithms, which took a lot of my time.

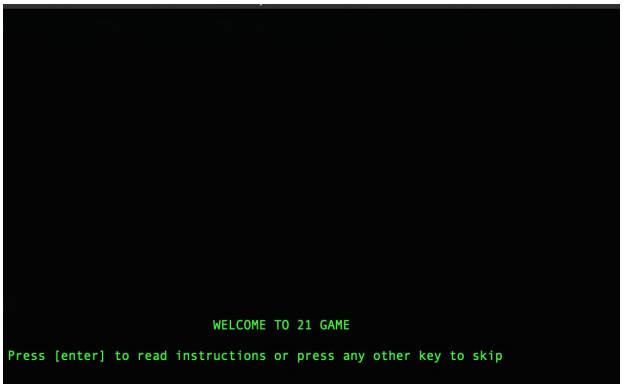
Lines of code: 936
Classes: 5
Github Link: https://github.com/angeldzzz23/TwentyOneGame

Description

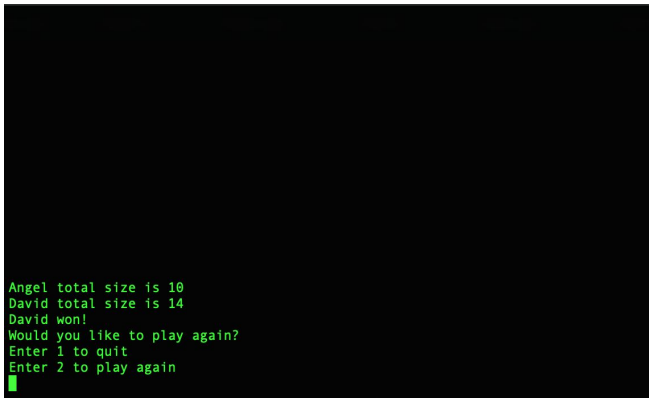
The objective of this game is to play a two player game. My game is organized in my twentyOne class.That class contains references to several other classes such as Cards, rank, and player. My card class consists of a rank and Suit, which are both other classes. In my twenty one class the game starts with the playTwoPlayerGame. My main function only has a few lines. I only call display instructions, initialGameState and playTwoPlayerGame.

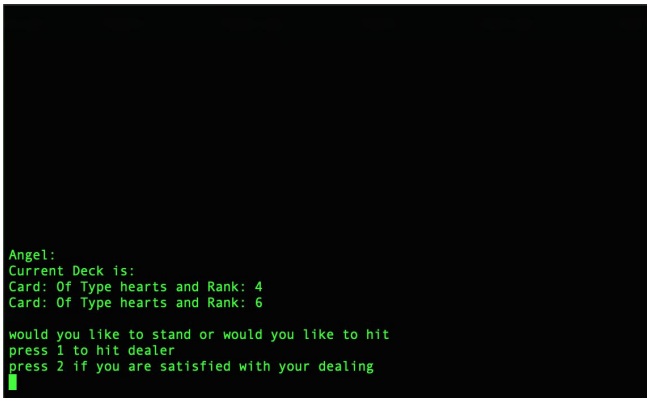
Sample input/output on how to play the game.

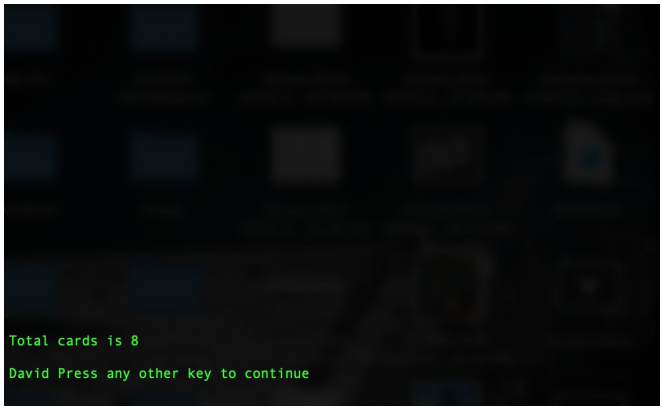
Tutorial



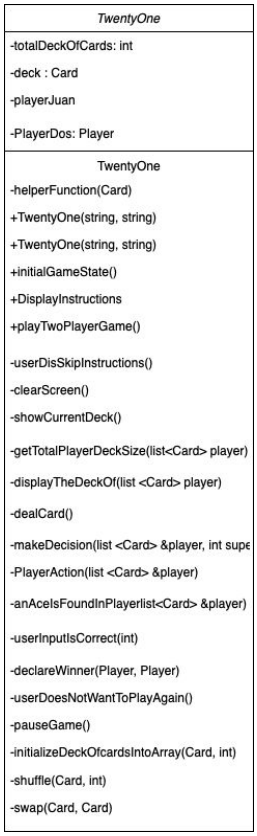
21 Game

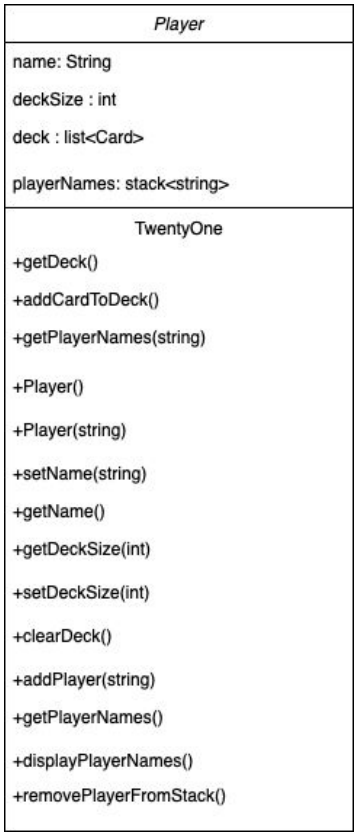


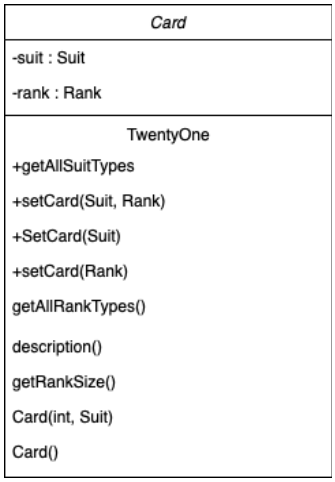


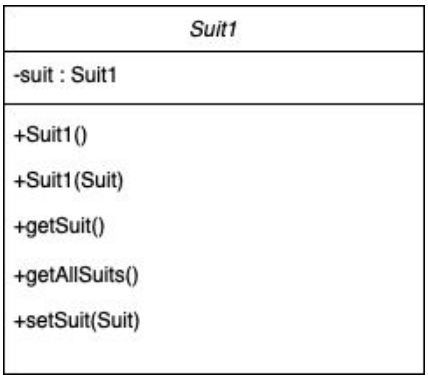


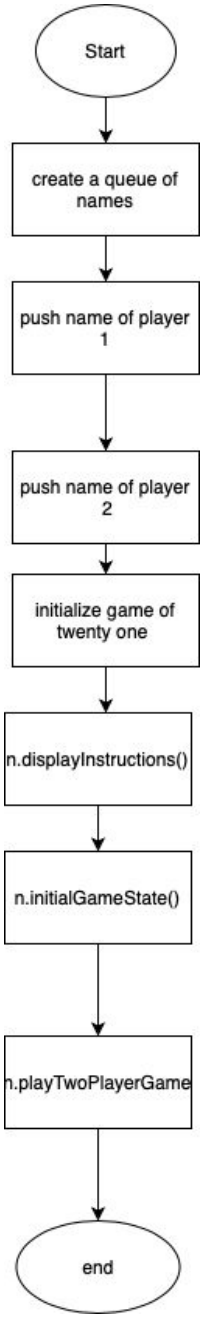
UML DIAGRAM (FINISHED)










Start
create a queue of names
push name of player 1
push name of player 2
initialize game of twenty one
n.displayInstructions()
n.initialGameState()
n.playTwoPlayerGame
end

CheckOffSheet

1. Container classes (Where in code did you put each of these Concepts and how were they used?
    1. Sequences (At least 1)
        1. List
            - I use this to create my deck of cards. You can find this in twentyOne Class.
        2. slist
        3. bit_vector
    2. Associative Containers (At least 2)
        1. set // Don't have a use for it
        2. map // Used in getRank func in the Rank class.
           I map special cases of ranks to their number (j,k,q)
           I use this in my rank class (getRank function).
        3. hash
    3. Container adaptors (At least 2)
        1. stack // Players name
           In the player class I used a stack to store the names of players.
        2. Queue
           // Used to add names into the initializers of twentyOne class in my main class.
        3. priority_queue
2. Iterators
    1. Concepts (Describe the iterators utilized for each Container)
        1. Trivial Iterator
        2. Input Iterator
        3. Output Iterator
           I used this output iterator with the list to print out the card description for the whole deck and individual players. I didnt really use iterators for my map or stack.
        4. Forward Iterator
        5. Bidirectional Iterator
        6. Random Access Iterator
3. Algorithms (Choose at least 1 from each category)
    1. Non-mutating algorithms
        1. For_each
           I use this to print out the description of cards in my TwentyOneClass in the displayCurrentDeck method.

2. find
3. count
4. equal
5. search

2. Mutating algorithms
    1. copy
    2. Swap

    I use this in my twentyOneClass in order to shuffle an array of cards
    3. Transform
    4. Replace
    5. fill
    6. Remove
    7. Random_Shuffle

3. Organization // Sorting A deck
    1. Sort
    2. Binary search
    3. merge
    4. inplace_merge
    5. Minimum and maximum //

    I used the maximum to find the max_element in my ranks.

```
// system level libraries
#include <iostream>
#include <list>
#include <iterator>
#include <queue>
#include <set>
#include <map>
using namespace std;  //Library Scope

//User Libraries
#include "TwentyOne.h"
#include "Card.h"

//Execution Starts Here
int main(void) {
  // creating a Queue to store names
  queue <string> names;
  names.push("Angel");
  names.push("David");
  // initialize a game of twentyOne
  TwentyOne n(names.front(), names.back());
  // start tutorial
  n.displayInstructions();
  // setting up initial game state
  n.initialGameState();
  // starting game
  n.playTwoPlayerGame();
// program ends
  return 0;
}
//Function Implementations


/*
 * File:   TwentyOne.h
 * Author: Angel Zambrano
 * Created on April 26, 2020, 12:43 AM
 * Specification for twentyOneGame Class
 */
```

```
#ifndef TWENTYONE_H
#define TWENTYONE_H

// system level libraries
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <ctime>       // std::time
#include <cstdlib>     // std::rand, std::srand

//User Libraries
#include "Card.h"
#include "Player.h"
#include "stdlib.h"


class TwentyOne {
private:
   list <Card> deck;
   Player playerJuan;
   Player playerDos;
   static void helperFunction(Card);
   bool userDidSkipInstruction();
   void ClearScreen();
   void displayCurrentDeck(); // prints out the current deck
   int getTotalPlayerDeckSize(list <Card> player); // gets the size of decks added of a
player
   void displayTheDeckOf(list <Card> player); // prints the deck of the user
   Card dealCard(); // returns a card from deck
   bool makeDecision(list <Card> &player, int userInput); //
   void playerAction(list <Card> &player);
   bool anAceIsFoundInPlayer(list <Card> &cards); // finds out if there is an ace on the
use's deck
   bool userInputIsCorrect(int input);
   void declareWinner(Player playerOne, Player playerTwo); // prints the winner of the
game
   bool userDoesNotWantToPlayAgain(); // finds out if the user wants to play again
   void pauseGame(); //
   void initializeDeckOfCardsIntoArray(Card *arr, int s);
   void swap(Card *a, Card *b); // swaps two variables
   void shuffle(Card *arr, int size);
public:
 TwentyOne(); // default constructor
```

```
  TwentyOne(string, string); //
  void initialGameState(); // sets card size to 52
  void playTwoPlayerGame(); // starts a game
  void displayInstructions();
 };
#endif

/*
 * File:   TwentyOne.cpp
 * Author: Angel Zambrano
 * Created on April 26, 2020, 7:46 AM
 * Specification for twentyOneGame Class
 */
#include <algorithm>

#include "TwentyOne.h"


TwentyOne::TwentyOne() { // Default constructor

}

// constructor that intializes playerNames
TwentyOne::TwentyOne(string playerOneName, string playerTwo)
:playerJuan(playerOneName), playerDos(playerTwo)
{

}

// Checks if user entered "\n"
bool TwentyOne::userDidSkipInstruction() {
  string str;
  getline(cin, str);

  if (str.length() == 0) {
      cout << endl << endl << endl;
      return true;
  }
  return false;

}

//   // prints out 100
```

```
 void TwentyOne::ClearScreen() {
    cout << string( 100, '\n' );
  }

  // initialize deck with 52 new cards
  // intialize totalDeck
  void TwentyOne::initialGameState() {

    // create a large array to store cards
    Card cardArray[55]; // s

    // clear deck if you are playig again
    if (!(deck.empty() && playerJuan.getDeck().empty() && playerDos.getDeck().empty()))
{
      // clear everything. Reset numbers
      deck.clear();
      playerJuan.clearDeck();
      playerJuan.setDeckSize(0);

      playerDos.clearDeck();
      playerDos.setDeckSize(0);

    }

    // creates array of deck
    initializeDeckOfCardsIntoArray(cardArray, 52);

      // shuffle
      shuffle(cardArray, 52);
      // initialize Deck with shuffled Cards
      for(int i = 0; i < 52; i++) {
        // append card into deck
        deck.push_back(cardArray[i]);
      }


    // initiaize each player with one card

    // initializing player one with two cards
    playerJuan.addCardToDeck(dealCard());
    playerJuan.addCardToDeck(dealCard());

    // intialize player two with two cards
```

```
  playerDos.addCardToDeck(dealCard());
  playerDos.addCardToDeck(dealCard());
}

// helps showCurrentDeck with printing the card
void TwentyOne::helperFunction(Card n) {
  n.description();
}

 // prints out deck using the iterator
 void TwentyOne::displayCurrentDeck() {
   // print each card in the deck
   for_each (deck.begin(), deck.end(), helperFunction);
 }


 // starts a two placer game when it starts
 void TwentyOne::playTwoPlayerGame() {
   // clear screen
   ClearScreen();
   // prints name of player 1
   cout << playerJuan.getName()<< ": " << endl;

   // prompt user with game choices
   playerAction(playerJuan.getDeck());
   // set player 1 deck size
   playerJuan.setDeckSize(getTotalPlayerDeckSize( playerJuan.getDeck()));
   // print out the total of cards player 1 has
   cout << "Total cards is " << playerJuan.getDeckSize() << endl << endl;

   // ask if player two is ready
    pauseGame();
    // clear screen
    ClearScreen();

    // show previous player's deck
    cout << "Last player "<<playerJuan.getName()<< ": " << endl;
    displayTheDeckOf(playerJuan.getDeck());
    cout << endl;

    // displays current player name
    cout <<playerDos.getName() << "(Current Player)"<< ": " << endl;
    // prompt user with game choices
```

```
      playerAction(playerDos.getDeck());
      // set player 2 deck size
      playerDos.setDeckSize(getTotalPlayerDeckSize( playerDos.getDeck()));
      // clear screan
      ClearScreen();
      // display both player's deck sizes
      cout << playerJuan.getName() << " total size is " <<  playerJuan.getDeckSize() <<
endl;
      cout << playerDos.getName()<< " total size is " <<  playerDos.getDeckSize() << endl;

      // display winner
     declareWinner(playerJuan,playerDos);

     cout << endl;
     // ask player if they want to play again
     if (userDoesNotWantToPlayAgain()) {
       // reset cards in initial game state
       initialGameState();
       // call playTwoPlayerGame
       playTwoPlayerGame();
     }
    }

   // displays the current deck of the player
   void TwentyOne::displayTheDeckOf(list <Card> player) {
    // iterator
    list <Card> :: iterator it;
    // displays each card in the player link list
    for(it = player.begin(); it != player.end(); ++it)  {
      it->description();
    }
   }

   // deals card to user
   Card TwentyOne:: dealCard() {
    // get the first card from the deck
    Card card = deck.front();
    // remove first card from deck
    deck.pop_front();
    // return the first card from the deck
    return card;
   }
```

```
// adds the player's deck size.
//also takes care of special cases such as J, K and Q as well as Ace
int TwentyOne::getTotalPlayerDeckSize(list <Card> player) {
  int userInput; // input can only be
  bool inputIsNotCorrect = true;

  if (anAceIsFoundInPlayer(player)) {

    // get input
    do {
      cout << "Would you like your ace to be 1 or 11" << endl;
      cout << "press 1 for 1"  << endl;
      cout << "press 2 for 11" << endl;
      cin >> userInput;

      inputIsNotCorrect = !userInputIsCorrect(userInput);

    }while(inputIsNotCorrect);
  }

  int size = 0;
  // card iterator
  list <Card> :: iterator it;
  // get the tota; of the deck size
  for(it = player.begin(); it != player.end(); ++it)  {
    if (it->getRankSize() == 1 && userInput == 2) {
      size += 11;
      continue;
    } else if (it->getRankSize() >= 11 && it->getRankSize() <= 13) {
      size += 10;
      continue;
    }
      size += it->getRankSize();
  }
  return size;
}

// the prompt of user input
bool TwentyOne::makeDecision(list <Card> &player, int userInput) {

  // if userInput is 1. Add a card into its deck
  if (userInput == 1) {
    // add card into player deck
```

```
    player.push_back(dealCard());

  }   // if userInput is 2. Then do nothing.
  else if (userInput == 2) {
    return false;
  }
return true;
}

// prompts the user with two choices
// hit or stand
void TwentyOne::playerAction(list <Card> &player) {
  bool playerIsNotDone =  true;
  int userInput;
  do {
    // print the cards user has
    cout << "Current Deck is: " << endl;
    displayTheDeckOf(player);
    cout << endl;

    // Ask user what they would like to do
    cout << "would you like to stand or would you like to hit" << endl;
    cout << "press 1 to hit dealer" << endl;
    cout << "press 2 if you are satisfied with your dealing" << endl;
    cin >>userInput;
    // call decision function
    playerIsNotDone = makeDecision(player,userInput);
    ClearScreen();

  } while(playerIsNotDone);

}

// loops through the place list if the
bool TwentyOne::anAceIsFoundInPlayer(list <Card> &cards) {
    list <Card> :: iterator it;
    for(it = cards.begin(); it != cards.end(); ++it) {
      if (it->getRankSize() == 1) {
        return true;
      }
    }
    return false;
  }
```

```
// finds out if user input is the correct range
bool TwentyOne::userInputIsCorrect(int input) {
  if (input <= 2 && input > 0) {
     return true;
  }
  return false;
}

// promps the user with a menu of playing again
bool TwentyOne::userDoesNotWantToPlayAgain() {
bool incorrectInput = true;
int userInput;
  // get userInput
  do {
    cout << "Would you like to play again?" << endl;
    cout << "Enter 1 to quit" << endl;
    cout << "Enter 2 to play again" << endl;
    cin>>userInput;
    // loops until userInput is correct
  }while(!(userInput <= 2 && userInput >= 1));

return (userInput ==  1) ? false : false;
}

// declares declareWinner
void TwentyOne::declareWinner(Player playerOne, Player playerTwo) {
  int p1size, p2size;
  int playerOneSize = playerOne.getDeckSize();
  int playerTwoSize = playerTwo.getDeckSize();

  // player one is 21
  if (playerOneSize == 21 && playerTwoSize != 21) {
    cout << "player 1 won" << endl;
    cout << playerOne.getName() << " won!";
  }
    // player two is 21
  if (playerTwoSize == 21 && playerOneSize != 21) {
    cout << playerTwo.getName() << " won!";

  }

  // tie
```

```
  if (playerTwoSize == 21 && playerOneSize == 21) {
    cout << "Both players tied" << endl;
  }

  // if both are not less than 21
  if (playerOneSize < 21 && playerTwoSize < 21) {

    if (playerOneSize > playerTwoSize) {
        cout << playerOne.getName() << " won!";
    }

    if (playerOneSize == playerTwoSize) {
      cout << "it is a tie!" << endl;
    }

    if (playerOneSize < playerTwoSize) {
      cout << playerTwo.getName() << " won!";
    }
  }

  if (playerOneSize > 21 && playerTwoSize > 21) {
    // might want to refactor this
    if (playerOneSize < playerTwoSize) {
      cout << playerOne.getName() << " won!";
    } else if (playerOneSize == playerTwoSize) {
      cout << "it is a tie!" << endl;
    } else if (playerTwoSize < playerOneSize) {
        cout << playerTwo.getName() << " won!";
    }
  }
}

// displays instructions
void TwentyOne::displayInstructions() {

    ClearScreen();
    cout << setw(45) << "WELCOME TO 21 GAME" << endl;
    cout << endl;
    cout << "Press [enter] to read instructions or press any other key to skip" << endl;

    if (userDidSkipInstruction()) {
       ClearScreen();
      cout << setw(30) <<"How to Play" << endl;
```

```
        cout << endl;
        cout << "* This is a two player game" << endl;
        cout << "* In each game you will be given 2 cards to start with" << endl;
        cout << "* You have the option to stand or hit the dealer"<< endl;
        cout <<endl;
        cout << "Press [enter] to reads more or any other key to skip" << endl;

        if (userDidSkipInstruction()) {
          ClearScreen();
          cout << setw(30)<<"SCORING & Tips"<< endl;
          cout << endl;
          cout << "* Which ever player reaches 21 wins" << endl;
          cout << "* Remember an ace can be either 1 or 11" << endl;
          cout << "* J Q and K are all counted as ten" << endl;
          cout <<endl;
          cout << "[Press any key to start a new game]"<< endl;

          if (userDidSkipInstruction()) {
             ClearScreen();
            return;
          } else {
             ClearScreen();
            return;
          }

        } else {
          ClearScreen();
          return;
        }
      } else {
        ClearScreen();
        return;
      }
}

// pauses the game
void TwentyOne::pauseGame() {
  string k;
  cout << playerDos.getName() << " Press any other key to continue" << endl;
  cin >> k;
}

// inttializes array into deck
```

```
void TwentyOne::initializeDeckOfCardsIntoArray(Card *arr, int s) {
  // set rank and suit variables to get access to their mehtods
  Rank rank;
  Suit1 suit;
  // set arrays of suit and ranks
  Suit *suits; // array of suit types
  int *ranks;

  // getting array of all suit types
  suits = suit.getAllSuits();
  ranks = rank.getAllRanks();

  // deck size
  int size = 0;
  // initialize deck into array

  // push new cards into deck
  for (int s = 0; s < 4; s++) {
    for (int r = 0; r < *max_element(ranks,ranks+13); r++ ) {
      // add card into deck
      // deck.push_back(Card(r+1, suits[s]));
      arr[size] = Card(r+1, suits[s]);
      // increase size of deck
        size++;
    }
    cout << endl;
  }
}

// swaps two variables by  referening
void TwentyOne::swap(Card *a, Card *b) {
  Card temp = *a;
  *a = *b;
  *b = temp;
}

// Shuffles the array
void TwentyOne::shuffle(Card *arr, int size) {
  srand (time(NULL));
  for (int i = size - 1; i > 0; i--)
  {
      int j = rand() % (i + 1);
      swap(&arr[i], &arr[j]);
```

```
    }
   }

/*
 * File:   Player.h
 * Author: Angel Zambrano
 * Created on April 26, 2020, 8:36 PM
 * PLayer header file
 */

#ifndef PLAYER_H
#define PLAYER_H
#include "Card.h"
#include <list>
#include <stack>

class Player {
private:
   string name; // name of player
   int deckSize; // total size of all deck added
   list <Card> deck; // deck
   static stack <string> playerNames; // stores all player names

public:
  list <Card>&getDeck(); // returns the address of deck
  void addCardToDeck(Card card); // adds card to deck
  void description(); // displays a descripcion of a card
  Player(); // default constructor
  Player(string); // constructor with name
  void setName(string); // sets name
  string getName(); // returns the name
  int getDeckSize(); // returns the total size of decks added
  void setDeckSize(int); // sets the size of the deck
  void clearDeck(); // clears deck
  void addPlayer(string); // add players into stack of players
  stack<string> getPlayerNames(); //
  void displayPlayerNames(); // displays all the player names
  void removePlayerFromStack();
};
#endif

/*
 * File:   Player.cpp
```

```
* Author: Angel Zambrano
* Created on April 26, 2020, 7:30 PM
* cpp file
*/

#include "Player.h"

// default constructor
Player::Player() {
  // gives player a default name
  setName("Default Name");
  addPlayer("Default Name");
}
// clears deck
void Player::clearDeck() {
  deck.clear();
}
// returns deck size
int Player::getDeckSize() {
  return deckSize;
}
// sets deckSize
void Player::setDeckSize(int _deckSize) {
  deckSize = _deckSize;
}

// returns the name
string Player::getName() {
  return name;
}
// sets the name
void Player::setName(string _name) {
  name = _name;

}
// default constructor
Player::Player(string _name) {
  name = _name;
  setName(name);
  addPlayer(name);
}
// returns the adress of deck
list <Card> &Player:: getDeck() {
```

```
  return deck;
}

// pushs deck to card
void Player::addCardToDeck(Card card) {
  deck.push_back(card);
}

// displays a descripcion of the player
void Player::description() {
  cout << getName()<< ": " << endl;
  // card iterator
  list <Card> :: iterator it;

  // prints out every element in the deck
  for(it = deck.begin(); it != deck.end(); ++it)  {
    it->description();
  }
  cout << '\n';
  cout << "Size of deck added: " << getDeckSize() << endl;
}

stack <string> Player::playerNames;

// appends player into stack of players
void Player::addPlayer(string _name) {
  // adds player to the top of the stack of player names
  playerNames.push(_name);
}

// return playerNames
stack<string> Player::getPlayerNames() {
  return playerNames;
}

void Player::removePlayerFromStack() {
 playerNames.pop();
}

// displays the players in a game
void  Player::displayPlayerNames() {
  stack <string> s =  playerNames;
```

```
  while (!s.empty())
  {
     cout << " " << s.top();
     s.pop();
  }
  cout << '\n';

 }

/*
 * File:   Card.h
 * Author: Angel Zambrano
 * Created on April 23, 2020, 8:12 PM
 * Specification for the Table
 */


#ifndef Card_h
#define Card_h
#include "Rank.h"
#include "Suit.h"
#include <string>
using namespace std;

class Card  {
private:
  Suit1 suit;
  Rank rank;
public:
  Suit* getAllSuitTypes();
  void setCard(Suit1 _suit, Rank _rank); // sets the card with new suit and rank
  void setCard(Suit1 _suit); // sets card with new suit
  void setCard(Rank _rank); // sets card with new rank
  string* getAllRankTypes();
  void description();
  int getRankSize();
  Card(int rankType, Suit suitType); // Constructor
  Card();

};
#endif /* TABLE_H */

/*
```

```
 * File:   TwentyOne.h
 * Author: Angel Zambrano
 * Created on April 26, 2020, 8:36 PM
 * Card methods implementation
 */


#include "Card.h"


Card::Card() {
  // DO Nothing
}

Card::Card(int rankType, Suit suitType)
: rank(rankType), suit(suitType)
{
    // Do Nothing
}

string* Card::getAllRankTypes() {
  string *ranks = new string[13];
  for (int i = 1; i <= 13; i++) {
    ranks[i - 1] = to_string(i);
  }
  return ranks;
}

void Card::description() {
  cout << "Card: " << "Of Type " << suit.getSuit() << " and Rank: " << rank.getRank() <<
endl;
}

Suit* Card::getAllSuitTypes() {

Suit*all = new Suit[4];
  all[0] = Spades;
  all[1] = Hearts;
  all[2] = clubs;
  all[3] = diamond;
  return all;
}
```

```
// returns the rank size
int Card::getRankSize() {
  return rank.getRankAsInt();
}

// sets the card with new suit and rank
void Card::setCard(Suit1 _suit, Rank _rank)
{
  suit = _suit;
  rank = _rank;
}

// sets card with new suit
void Card::setCard(Suit1 _suit) {
suit = _suit;
}

// sets card with new rank
void Card::setCard(Rank _rank){
  rank = _rank;
}

/*
 * File:   TwentyOne.h
 * Author: Angel Zambrano
 * Created on April 26, 2020, 8:36 PM
 * Suit Class
 */

#ifndef SUIT_H
#define SUIT_H
#include <iostream>

enum Suit {Spades = 1, Hearts, clubs, diamond};
class Suit1 {
private:
  Suit suit;
public:
  Suit1(Suit);
  Suit1();
  std::string getSuit(); // get current suit
  int getTotalSuits(); // returns the int size
  Suit* getAllSuits(); // returns an array of all the types of suits
```

```
  void setSuit(Suit); // ses the suit

};
#endif

/*
 * File:   TwentyOne.h
 * Author: Angel Zambrano
 * Created on April 26, 2020, 8:36 PM
 * Specification for the Suit
 */

#include "Suit.h"
using namespace std;

Suit1::Suit1() {
    // TODO
}
Suit1::Suit1(Suit _suit) {
  suit = _suit;
}
// return the possible suit
int Suit1::getTotalSuits() { return 4; }


// returns current suit
string Suit1::getSuit() {
  switch(suit) {
    case 1:
      return "spades";
      break;
    case 2:
      return "hearts";
      break;
    case 3:
      return "clubs";
      break;
    case 4:
      return "diamonds";
      break;
    default:
      return "ppp Something went wrong";
      break;
```

```
  }
}

// retruns an rray of suits
Suit* Suit1::getAllSuits() {
  Suit*all = new Suit[4];
  all[0] = Spades;
  all[1] = Hearts;
  all[2] = clubs;
  all[3] = diamond;
    return all;
}

// sets the suit
void Suit1::setSuit(Suit n) {
  suit = n;
}

/*
 * File:   TwentyOne.h
 * Author: Angel Zambrano
 * Created on April 26, 2020, 10:41 PM
 */

#ifndef RANK_H
#define RANK_H
#include <iostream>
#include <string>
#include <list>

using namespace std;

class Rank {
private:
  int rank;
  void rangeError(); // displays and extis with an error
  void helperFunctionForDisplayingRanks(int i);
public:
  Rank(int n); // constructor
  Rank(); // default contrustor
  int getAmountOfRanks(); //
  string getRank(); // retruns current rank
  int *getAllRanks(); // retruns an array of rank ints
```

```
  int getRankAsInt(); // returns the rank
  void setRank(int n);
  void displayRanks();
};
#endif

/*
 * File:   Rank.h
 * Author: Angel Zambrano
 * Created on April 24, 2020, 10:41 PM
 */

#include "Rank.h"
#include <map>
// this is the total amount of ranks
//that can be in a card
int Rank::getAmountOfRanks() { return 13;}

Rank::Rank() {  } //

Rank::Rank(int n) {
  // Check if it is out o range
  if (n > getAmountOfRanks() || (n <= 0)) {
    rangeError();
  }
  // initialize rank
  setRank(n);
}

string Rank::getRank() {
      // dictionary to map int to numbers
     map<int, string> numbers;

     // create key value pairs
     numbers.insert(pair<int, string>(1, "A"));
     numbers.insert(pair<int, string>(11, "J"));
     numbers.insert(pair<int, string>(12, "Q"));
     numbers.insert(pair<int, string>(13, "K"));

  if (rank == 1) {
    return numbers[1];
  }
  else if (rank >= 1 && rank <= 10) {
```

```
    return to_string(rank);
  }
  else { // Face
    if (rank == 11) {
        return numbers[11];
      } else if (rank == 12) {
        return numbers[12];
      } else if (rank == 13) {
        return numbers[13];
      }
  }
  return "YOU ARE WRONG!";
}

// exit if it is out of range
void Rank::rangeError() {
  cout << "ERROR: out of range " << endl;
  exit(EXIT_FAILURE);
}

// returns an rray of rank ints
int *Rank::getAllRanks() {
  int *arrayp = new int[13];

  for (int i  = 0; i < 13; i++) {
    arrayp[i] = i + 1;
  }
  return arrayp;
}

  // returns the rank
  int Rank::getRankAsInt() {
    return rank;
  }
  // sets the rank
  void Rank::setRank(int n) {
    rank = n;
  }

  void helperFunctionForDisplayingRanks(int i) {  // function:

    if (i == 1) {
      cout << " "  << "A";
```

```
  } else if (i == 11) {
    cout << " " << "J";
  } else if (i == 12) {
    cout << " " << "Q";
  } else if (i ==  13) {
    cout << " " << "K";
  } else {
    cout << ' ' << i;
  }
}

void displayRanks() {
  // create a ranks list
  list<int> ranks;
   //push numbers into ranks
   for (int i=1; i<=13; ++i) ranks.push_back(i);
   for_each (ranks.begin(), ranks.end(), helperFunctionForDisplayingRanks);
}
```