# Lab #10 - Prof. Annexstein CS 1021C-002 - Spring 2014

## Due: Try to complete this lab assignment and submit it via CascadeLMS during the lab session on Wednesday, March 26. If you do not submit it during the lab session, you must submit it via CascadeLMS by the day of the next lab on Wednesday, April 2.

## Objective: Gain experience and practice with classes and objects, and designing member functions.

### Problem Description: You are to implement and modify a set of C++ classes for modeling a DeckOfCards.

**Task 0.**

In this initial part you will download and execute the code for the Card Dealing Simulation and the DeckOfCards. A copy of this code is found at the end of this document.

**Task1.**

In this part you will modify and add a member function to the DeckOfCards class. Add a member function called dealNextCard(), which returns the next card on top of the deck. You will also need to keep track of the top of the deck. You also will need to implement a new class called Card that is used to create card objects. The member function dealNextCard() should return a Card object.

Test your code by creating an application that shuffles a deck and deals and prints the top 5 cards.

**Task 2.**  
in this part you will create a new class called Hand that is used for objects that model a 5-card poker hand. Thus, a Hand object should store 5 arbitrary card objects. Modify your test application so that it shuffles and deals two 5-card hands, alternating cards between hands as the cards are dealt.

**Task 3.**  
Write a member function called evaluate() for the Hand class. This function should return a boolean value and determine whether or not a hand object contains a pair of cards with the same face value. Test this code by evaluating each of the hands that are dealt.

### Task 4. (extra credit)

### Write an auxiliary function that takes two hand objects as parameters and returns which of the two hands is better. A full version of this function is quite challenging. See how accurate you can make it in the time permitted.

// deckOfCards.cpp

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

using namespace std;

// DeckOfCards class definition

class DeckOfCards

{

public:

DeckOfCards(); // constructor initializes deck

void shuffle(); // shuffles cards in deck

void deal(); // deals cards in deck

private:

int deck[ 4 ][ 13 ]; // represents deck of cards

}; // end class DeckOfCards

// DeckOfCards default constructor initializes deck

DeckOfCards::DeckOfCards()

{

// loop through rows of deck

for ( int row = 0; row <= 3; row++ )

{

// loop through columns of deck for current row

for ( int column = 0; column <= 12; column++ )

{

deck[ row ][ column ] = 0; // initialize slot of deck to 0

} // end inner for

} // end outer for

srand( time( 0 ) ); // seed random number generator

} // end DeckOfCards default constructor

// shuffle cards in deck

void DeckOfCards::shuffle()

{

int row; // represents suit value of card

int column; // represents face value of card

// for each of the 52 cards, choose a slot of the deck randomly

for ( int card = 1; card <= 52; card++ )

{

do // choose a new random location until unoccupied slot is found

{

row = rand() % 4; // randomly select the row (0 to 3)

column = rand() % 13; // randomly select the column (0 to 12)

} while( deck[ row ][ column ] != 0 ); // end do...while

// place card number in chosen slot of deck

deck[ row ][ column ] = card;

} // end for

} // end function shuffle

// deal cards in deck

void DeckOfCards::deal()

{

// initialize suit array

static const char \*suit[ 4 ] =

{ "Hearts", "Diamonds", "Clubs", "Spades" };

// initialize face array

static const char \*face[ 13 ] =

{ "Ace", "Deuce", "Three", "Four", "Five", "Six", "Seven",

"Eight", "Nine", "Ten", "Jack", "Queen", "King" };

// for each of the 52 cards

for ( int card = 1; card <= 52; card++ )

{

// loop through rows of deck

for ( int row = 0; row <= 3; row++ )

{

// loop through columns of deck for current row

for ( int column = 0; column <= 12; column++ )

{

// if slot contains current card, display card

if ( deck[ row ][ column ] == card )

{

cout << setw( 5 ) << right << face[ column ]

<< " of " << setw( 8 ) << left << suit[ row ]

<< ( card % 2 == 0 ? '\n' : '\t' );

} // end if

} // end innermost for

} // end inner for

} // end outer for

} // end function deal

void main() {

DeckOfCards mydeck;

mydeck.shuffle();

mydeck.deal();

return;

}