**CS161 - Programming Assignment - CardSuperFun**

The purpose of this assignment is to give you some practice working with C++ arrays.

This program will be the first of two that will implement the card game "21" (blackjack). In this part we use an array to implement a deck of cards (each "card" is an int) and we create, shuffle, and display them.

---

**Data (cards) representation and manipulation**

In the next assignment we're going to write a program to play the game 21 (blackjack). To do so, we'll need a virtual deck of cards for our game, but there is no "card" or "deck" datatype, nor do we currently have the knowledge to make one (no comedy intended, because we will by the end of the next term).

So, we'll use an array of integers for the deck, with each integer representing one card. For a single card, the 100's digit will designate the suit (Hearts, Diamonds, Clubs, Spades), and the tens and ones digit designating the Rank (Ace, Two, Three, Four, … Jack, Queen, King)

For example:

101 = Ace of Hearts
102 = Two of Hearts
…
110 = 10 of Hearts
111 = Jack of Hearts
112 = Queen of Hearts
113 = King of Hearts
201 = Ace of Diamonds
202 = Two of Diamonds.
...

---

**Program Implementation**

<u>Details</u>

Your program will consist of a main() which declares an array of 52 integers (the deck) and then calls functions to create, shuffle, and display the cards.

The routines to be written are described below. After writing a routine, test it thoroughly to verify it works properly before moving on. This sometimes requires additional code that won't be used in the final project (such as DumpDeck), but it's good practice to test each routine thoroughly before moving on, as it's easiest to debug and test something immediately after you've written it since it's fresh in your mind at that time.

**`void initializeDeck( int deck[]);`**

initializes the given (52 integer) array to hold 52 “cards” as per the example above. Note: size isn’t an argument because deck must be 52 cards.

**`void dumpDeck(const int deck[], int size);`**

displays the given deck as integers one per line (used for testing just to verify initialzeDeck worked).

**`void shuffle(int deck[], int size);`**

randomly rearranges the cards in the given 52-card deck. (this can be easily done by randomly selecting two cards and swapping them. Then repeating that 500 times or so. See the addendum at the bottom for a getRandomNumber routine you can cut and paste into your program).

**`void showCard(int card);`**

displays the given card’s rank and suit (on one line, with no space between them, and no endl after them). The rank will be A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, or K. The suit will be displayed using the ASCII character values 3, 4, 5, 6, which print out the card symbols (heart, diamond, club, and space) when using the characters set used by our compiler. (e.g, 101 should display as “A♡”.

```
//demo code
cout << static_cast<char>(3); //prints a heart symbol
```

**`void showCards(const int cards[], int numCards, bool hideFirstCard);`**

displays each card in the given array (deck or hand), space delimited, as described in showCard. (just make a call to showcard for each). Elements that are “cards” are those with non-zero values. So an array with the values 104, 410,103, 0, 0, would display 4♡ 10♠ 3♡. The last two zeros aren’t cards and won’t display.
If hideFirstCard is true, display two ** in place of the card.

---

**Addendum** - getRandomNumber() routine (since this wasn’t covered in class).

```
//==============================================================
// Function:    getRandomNumber
// Description: returns a random number between given low and high
//     values, inclusive.
//     Note: include cstdlib (for rand) and ctime (for time).
// Arguments:
//     low (I) - The lowest number to be generated
//     high (I) - The highest number to be generated (must be > low)
// Return value:
//     A random number between low and high (inclusive)
```

```
//=================================================================
int getRandomNumber(int low, int high) {
	static bool firstTime=true;
	int randNum;

	//if first time called, seed random number generator
	if (firstTime) {
		srand( static_cast<unsigned int>(time(NULL)) );
		firstTime=false;
	}

	//generate random number between given low and high values (inclusive)
	randNum = rand() % (high-low+1) + low;

	return randNum;
}
```