CS7A – Spring '13 – Lab 10

Here is the code we developed for handling a standard deck of cards.

```
// G. Hagopian
// Crazy Eights

#include <iostream>
#include <ctime>
#include <cstdlib>

using namespace std;

void shuffle(int cards[]);
void showCards(int cards[]);
void swap(int& first, int& second);

int main()
{
      //build a deck of cards, shuffle them and show them
      int cards[52];
      for(int i = 0; i < 52; ++i)
            cards[i] = i;
      shuffle(cards);
      showCards(cards);
}

void swap(int& first, int& second)
{
      int temp = second;
      second = first;
      first = temp;
}

void shuffle(int cards[])
{
      swap(cards[2], cards[7]);
}

void showCards(int cards[])
{
      for(int i = 0; i < 52; ++i)
            cout << "\nrank = " << cards[i]%13+1 << " suit = "
                 << cards[i]/13+1;
}
```

The output below shows the shuffle is minimal. Only the 3rd and the 8th cards have been swapped.

```
rank = 1 suit = 1
rank = 2 suit = 1
rank = 8 suit = 1
rank = 4 suit = 1
rank = 5 suit = 1
```

```
rank = 6 suit = 1
rank = 7 suit = 1
rank = 3 suit = 1
rank = 9 suit = 1
rank = 10 suit = 1
……
……
rank = 9 suit = 4
rank = 10 suit = 4
rank = 11 suit = 4
rank = 12 suit = 4
rank = 13 suit = 4
```

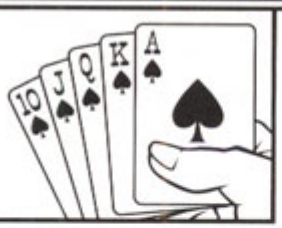
Your assignment is to

1. Use "perfect shuffle" of problem 5.11 to shuffle the cards. How many of these perfect shuffles are required to return the deck to its original state?
2. Create a more robust shuffle to randomize the order of the cards.
3. Once the cards have been shuffled to satisfaction, deal out two 5-card "hands," one for the "player" and one for the "dealer," and rank them according to the chart shown below. Write code to determine and report whether the player or the dealer have won. Note that the deck is finite here, so a hand can never have the same card twice.
4. Implement a "draw" feature into your code that will allow both the "player" to swap up to three cards from the depleted deck (at the user's discretion) and the "dealer" to swap up to three cards, either randomly or (extra credit!) using some AI. The computer should first prompt the "player" for how many and which cards to swap and then deal these from the remaining cards in the deck, and then do the same thing for the "dealer." Now determine which of these hands wins according to the rankings shown below. Use arrays and functions appropriately throughout.

POKER HAND RANKINGS

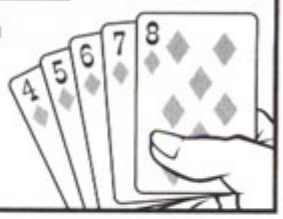
1. Royal Flush

A, K, Q, J, 10 all of the same suit



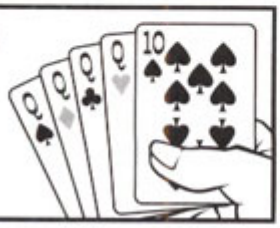
2. Straight Flush

Any five card sequence in the same suit



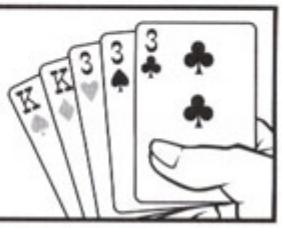
3. Four of a Kind

All four cards of the same rank



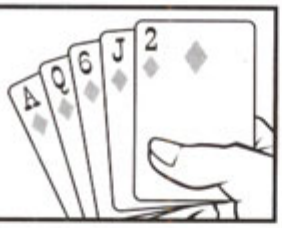
4. Full House

Three of a kind combined with a pair



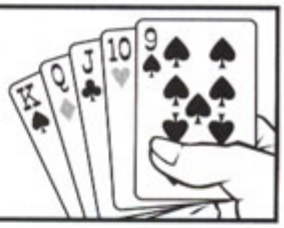
5. Flush

Any five cards of the same suit, but not in sequence



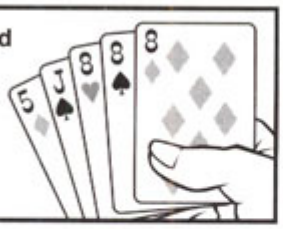
6. Straight

Five cards in sequence, but not in the same suit



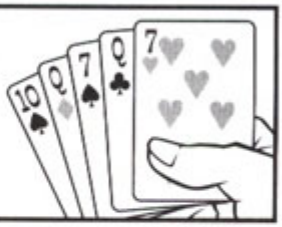
7. Three of a Kind

Three cards of the same rank



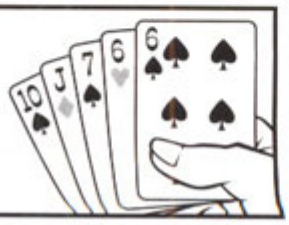
8. Two Pair

Two separate pairs



9. Pair

Two cards of the same rank



10. High Card

Otherwise unrelated cards ranked by the highest single card