War Game Assignment

This assignment will help you to review concepts from ICS 3U, as well as learn some differences in syntax between Java and C++. You will be creating a version of the card game War. Here are the setup and rules of this version.

War is played between 2 players. The original version involves players flipping over the top card of their piles simultaneously, comparing the number on each card and the player with the higher number adds both cards to their personal discard pile. When each player runs out of cards, they shuffle their discard pile to make their new deck. A player loses if they run out of cards in their deck and their discard pile is empty.

We are going to add some modifications to the basic rules.

1. Instead of flipping over the top card, both players draw 4 cards from their pile, look at them, and choose the order to play them in.

2. Once each player has chosen, cards are then compared in the order they chose, and the winner of each "battle" takes the cards as normal (they go to that players discard pile). Twos are low (equal to 2), Aces are high (equal to 14).

3. In the event of a numerical tie, suit is the tiebreaking factor. Suits are listed from strongest to weakest as follows: Clubs, Diamonds, Hearts, Spades (notice they are alphabetical).

4. *Players now lose if they* are unable to draw 4 cards from their pile at the start of their turn (including cards in discard pile, which would be shuffled in if needed). Shuffling should only happen when a players pile drops below 4 cards (not each time the player wins cards).

Given:

Two *constant* (unchangeable) string arrays called faces and suits that contain the following values:

faces : "Ace", "Deuce", "Three" ,"Four", … "Ten", "Jack", "Queen", "King".
suits: "Hearts", "Diamonds", "Clubs", "Spades".

Create a main function that does the following:

a) Create a full deck of cards (52). Each card should be of the form "<face> of <suit>". You should create and initialize this in the main function, though I want you to first try to create and return the array from a helper function and see what happens. This is an important difference to observe. You may also discover a workaround while doing this.
b) Shuffle the deck of cards.
c) Deal the deck into two separate piles.
d) Begin the game.

Be sure to make effective use of input and output. When comparing cards, you will need a way to extract the necessary information from the card each time (ex. "Jack of Hearts" will have a numerical value of 11 for comparison, and the suit "Hearts" is the third highest ranked in the event of a tie.

When a players pile runs out, you will need a way to shuffle the discard back into the players pile, and your program will need to be able to check if a player has won.

Additional Criteria:

Make sure to use a “boolean” variable at least once in the program.
Use helper function(s) *at least once* in the program in a meaningful way. There should be multiple opportunities to do this.

Level Achievement Criteria
up to 2 (max 70%)
- creates a deck
- shuffles deck, splits into piles and deals a hand to each player. Game is working to some extent.
- obvious functions included with some of the required combinations working

up to 3 (max 80%)
- creates a deck correctly
- all criteria from Level 2, along with a fully functioning play system
- functions used to simplify and/or organize program

up to 4 (max 90%)
- exceptional use of functions to simplify and/or organize program
- good use of string manipulation and arrays
  - functions are efficient and effective

up to 4+ (max 100%)
- additional game-like features
  - game-like environment with functioning methods that successfully determine winner