CSI 201
Functions - Putting things together - more practice

1. Here are the functions I want to write today with a brief description of each. But to start, let's make sure we know a little about what we're trying to implement. We want to implement a random card draw. So here's some information: //A card in a deck of card has a suit and a face. There are 4 suits and 13 faces for 52 cards total
   //There are 4 suits of cards:
   //0-12 in this deck are "spades"
   //13-25 in this deck are "clubs"
   //26-38 in this deck are "diamonds"
   //39-51 in this deck are " hearts"
   //in each suit there are 13 faces denoted by: "A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K"

   (a) int playingCard(); //this function will get a random number 0 - 51 that represents a card in a deck of cards
   (b) int suit(int); //this function will return 0, 1, 2 or 3 representing the "suit" of the card given a number 0-51
   (c) string suitString(int); //this function will return "spades", "clubs", "diamonds", "hearts" based on the face 0,1,2,3 of the card
   (d) int face(int); //this function will return 0-12 representing the face of the card given a number 0-51
   (e) string faceString(int); //given 0 -12, this function returns the string representing the face of the card
   (f) int faceValue(int); // given 0-12 representing the face of a card return its value. In this case, treat A as 1, 2-10 as 2-10, and K Q J as 10.
   (g) void displayCard(int); //this function displays a card given 0-51
   (h) int higherCard(int, int); //given two numbers 0-51 for cards, this function returns a 1 if the face of the first card is better than the face of the second card. In this case, A is the best face, followd by K. Full ordering is: "A K Q J 10 9 8 7 6 5 4 3 2"
   (i) int sumCard(int, int); //find the sum of the faces of two cards (0-51 values). So if you're adding K hearts and A spades the sum is 11.

2. Lastly, unit tests are something that are required in industry. Unit tests are code written that test that other code works as desired. In a sense, the labs that are in zybooks are also kinds of unit tests that are testing our programs and functions.

3. In addition `#include <cassert>` will let us use an assert function. assert allows us to test if a condition is true (or not) and to "crash" your program if it is not. For example, we might want to make the assertion that our face function from above only returns a value from 0 to 12. So we might throw a bunch of test conditions that assert that is true. Imagine the following code:

```
for(int i = -100; i <= 100 ; ++i) {
    assert(face(i) >= 0 && face(i) <= 12);
}
```

4. This code will assure that any "givens" to the function from -100 to 100 will return a value from 0 to 12. Of course, some of those function parameters are "improper." But this is why unit tests can be useful. They help us define what should happen in error cases as well.

5. They help guide our expectations. Our project managers might be writing some unit tests, or that might be left up to us. But either way, they will help us verify that our code is working as desired.