# Assignment #6: Blackjack Redux!

Once again you will be developing a blackjack application, but this time you'll be doing it in C++. The structure is exactly the same as last time:

In this assignment, you will build a simple game of blackjack where the computer plays as the dealer. In this simplified version of blackjack, the active player is dealt a card whose value ranges between 2 and 11. For simplicity, we will ignore the dual role that aces play (either as 1 or 11) and suits. Because of this, some impossible situations may occur, such as 5 2-value cards being dealt in a given hand. Likewise, this game does not use true odds as we won’t be modifying the odds of getting a card based on previous cards dealt.

## Program Flow

Because C++ is more procedural in its execution, program flow will be a little different than your even-based JavaScript version:

### Program Start

1. Output an introduction

### Starting with the player, do the following

1. Deal the player a card.
2. Add the card’s value to the player’s total hand.
3. Output the value of the card just dealt and the player’s current hand total (score)
4. Ask the player if he would like to hit or stand
5. If the player hits, go back to #1

### After the player busts (goes over 21) or stands

1. If the player has not busted, begin computer’s play
2. Deal the computer a card
3. Add the value of the card to the computer’s hand total (score)
4. If a card causes the computer to bust (go over 21), output a message. Go to step #7.
5. If the computer’s hand total is less than 17, it must take another card (hit). Go to step #2.
6. If the computer’s hand is 17 or more, the computer stands.
7. Stop computer play

### When both player and computer are done

1. If the player has not busted (gone over 21) and the computer busted, the player wins
2. If the player has not busted, the computer has not busted, and the player’s hand total is greater than the computer, then the player wins
3. If the player has not busted, the computer has not busted, and the player’s hand total is less than the computer’s then the computer wins
4. If the player has busted, then the computer wins.
5. Ask the player if he would like to play again.

## Recommended Variables

It might be helpful for your program to track the following items using variables:

* Whether or not the player would like to play again
* Whether or not the player would like to hit or stand
* The player’s current score
* The computer’s current score
* Whether or not the player has busted
* The value of the next card drawn

## Required Functions

In this function, you must define and use the functions described below. Note that it's okay to create more functions than what I have listed!

### int getNextCard()

* Randomly returns a value between 2 and 11
* (bonus, see grading) In addition to a value, your program returns and displays a valid suit (**NOTE**: this will require you to change the return type of the function!)
* (bonus, see grading) In addition to a value and suit, your program uses true odds
  + E.g. when an Ace of Diamonds is dealt, it is guaranteed not to be dealt again for the remainder of the game
* (bonus, see grading) In addition to the above two bonuses, your program allows the user to play with multiple decks and maintains true odds. Example, when playing with two decks, it is possible to draw up to two Ace of Diamonds per game, but no more.

### void playGame()

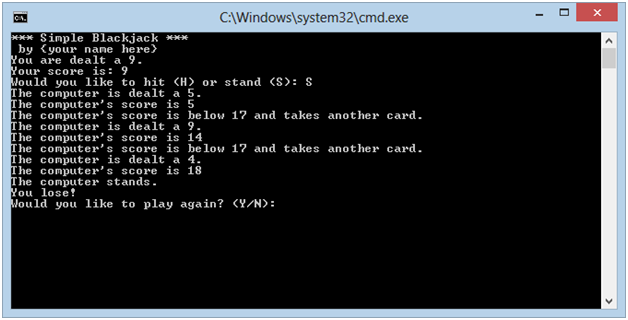
* Begins play of the game. Will call other functions as necessary.

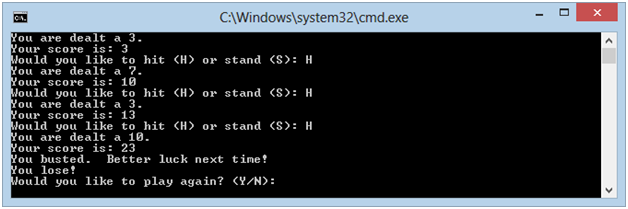
### int main(void)

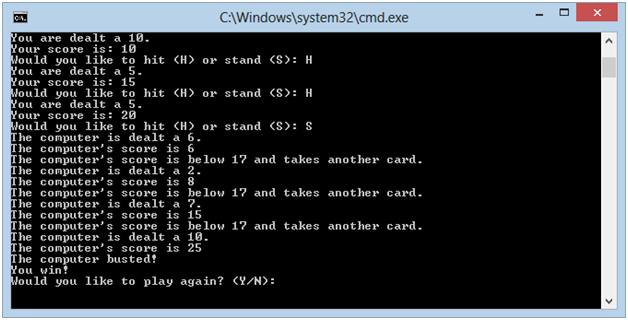
* Outputs welcome message
* Calls the play\_game function.

### Sample Output

The three screens below demonstrate possible play scenarios.







## Deliverables

You must upload your program through Canvas no later than midnight on Friday, November 16, 2018. In addition, you will demo the current state of your code during lab 11. This demo will be for credit.

## Grading Criteria

Your assignment will be judged by the following criteria:

### Proper Implementation

* [10] The program runs without crashing
* [10] The program randomly draws cards
* [**bonus** **3**] In addition to a value, your getNextCard () function returns and displays a valid suit
* [**bonus 4**] In addition to a value and suit, your getNextCard() function uses true odds
* **[bonus 3]** In addition to the above two bonuses, your getNextCard() allows the user to play with multiple decks and maintains true odds. Example, when playing with two decks, it is possible to draw up to two Ace of Diamonds per game but no more.
* [5] The program uses a loop to prompt the user for continual plays
* [10] The program correctly identifies when the player has busted
* [10] The program correctly identifies when the computer has busted
* [10] The program correctly identifies the winner
* [15] The computer always hits when its score is less than 17 and always stands when the score is 17 or greater

### Lab #11 Checkin

* [10] Your demo during lab #11 indicates that you are on track to receive full credit on the assignment.