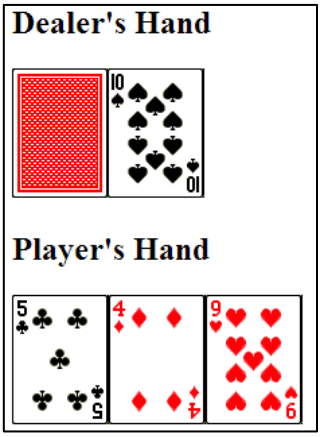
CS 105

Lab #6: Hit or Stand?
February 26–28, 2014



In lab this week, you will program a portion of a popular card game called **blackjack** (sometimes called **twenty-one**). Blackjack is played with a standard deck of playing cards: fifty-two cards divided up into four suits (clubs, diamonds, hearts, and spades) each with thirteen ranks (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, and K).

The goal of a game of blackjack is to have a hard that has as close to 21 points, without going over. In standard blackjack, an ace (rank "A") is worth either 1 or 11 points. **For simplicity, we will consider all aces to be worth only one point.** Beyond that, the cards 2-10 are worth the rank in point value. Jacks (J), Queens (Q), and Kings (K) are each worth 10 points. For blackjack, the suit of the card is irrelevant.

In this lab, our game of blackjack will have only two players: a **player** and a **dealer**. We have already provided you with a base set of code to allow you to play as the player; you will need to program the code to determine if the dealer should take another card. Luckily, the dealer follows a very simple set of rules:

- If the dealer has 16 or fewer points, the dealer must take another card ("hit").
- If the dealer has 17 or more points, the dealer must **not** take another card ("stand").

Understanding the Dealer Function

Inside the provided code, you will find `lab6.js`. This file contains one function that you must complete: `dealerHit(hand)`. The one and only parameter to the `dealerHit()` function, `hand`, is an array of strings that represents the dealer's current hand. Each string within the array is formatted to represent one of the 52 possible cards and is formatted in the following way: `<Suit><Rank>`, where `<Suit>` is either C, D, H, or S for the four suits and `<Rank>` is A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, or K for the thirteen ranks. For example, the string "C4" represents the four of clubs, "SQ" represents the queen of spaces, and "D10" represents the ten of diamonds.

Since `hand` is an array of these strings, it will be necessary to count the total number of points across all of the cards in the hand. A few examples are given below:

Data in `hand`	Total Point Value	Return Value
`["C7", "H4"]`	11	`true`
`["DA", "C8", "C4"]`	13	`true`
`["HK", "HQ"]`	20	`false`
`["C9", "C2", "D5", "H5"]`	21	`false`
`["C5", "CA"]`	6	`true`

As implied by the function name and the table above, the function should return `true` when the dealer should "hit" (to take another card) and `false` when the dealer should "stand". The introduction of this lab explains the specific rules of when the dealer should hit or stand.

Before you start coding, you should work out the pseudo-code to this algorithm. We have given you some hints on the back of the paper, but try and work out the code before checking the back of this paper!

The lab continues on the reverse side of this sheet of paper.

Programming the Dealer Function

When you are beginning to think of how to code the solution, you and your team should have identified two key problems that you need to address in your code:

- To count the total point value, you must look at every string inside an array. How do we iterate through an array? Would this be a **`for`**-loop?
- How do we get the rank of the card? Would **`charAt()`** be helpful?

One possible solution to solve this week's lab has been outlined in **pseudo-code** below:

```
set points to 0

for each string in the hand array:
   set rank to be the second character in the string

   if rank is "A": add one to points
   if rank is "2": add two to points
   if rank is "3": add three to points
   ...

if points is less than 17:
   return false
else:
   return true
```

…as with most programming problems, this is just one solution. If you thought of an alternative solution, you certainly should try it out!

Either using the algorithm we provided in pseudo-code or one your team came up with, you will need to translate this logic and program it in JavaScript. Once you have completed this, your game should be fully functional – go play some blackjack!

!	*If your game appears to work, you might have correctly programmed your code. However, this lab was designed in a way that **it is very likely you have a bug in your code** if you were not careful. Make sure to play several games before submitting your code. Specifically, make sure that a 10 appears in a few hands of the dealer.*

Submitting the Lab

As usual, submit the lab via the CS 105 website. Remember, **every person in your group must submit the lab**. In the past, we have been generous with grading the labs. As the semester continues, we will actually be choosing one random member and their program **will** be the grade for everyone. It is your job to make sure everyone who is in your group who is in lab has a complete program!