Problem (8 points, 12 minutes): Celebrity poker needs programmers like you.

Write `card-greater?` The predicate takes two cards and returns true if and only if the first card is bigger than second.

Cards are represented by a two-character word, where the first character represents the rank (`a`, `k`, `q`, `j`, `0`, `9`, `8`, `7`, `6`, `5`, `4`, `3`, and `2`), and the second character represents the suit (`s`, `h`, `d`, and `c`). For instance, `2h` is the two of hearts, `qc` is queen of clubs, `0s` is the 10 of spades, etc. For this problem, consider *all* spades to rank higher than hearts, which all rank higher than diamonds, which all rank higher than clubs.

```
(card-greater? 'ac '3d) ➔ #f
(card-greater? 'kh 'qh) ➔ #t
(card-greater? '4s '4s) ➔ #f
```

Comment all your procedures. Assume you have a working version of `outranks?`, as you wrote in lab, to use. (Remember, `outranks?` takes two ranks and returns true if the first is higher than the second.)

You need to use proper abstraction. In this case, you will need to define accessors, name them meaningfully, and include comments indicating their purpose.

There were a variety of ways you could go about this problem. We were strict about requiring you to use accessors for the suit and rank of a card, but were not as strict about strange constructions of the conditional. Here is a nice solution:

```
;; Returns true if the second card outrans the first
(define (card-greater? card1 card2)
   (if (equal? (suit card1) (suit card2))
       (outranks? (rank card1) (rank card2))
       (outsuits? (suit card1) (suit card2))))

;; Accessor: gets the rank of the given card
(define (rank card)
   (first card))

;; Accessor: gets the suit of the given card
(define (suit card)
   (first (bf card)))

;; Returns true if the first suit beats the second
(define (outsuits? suit1 suit2)
   (> (suit2number suit1) (suit2number suit2)))

;; Returns the rank of the suit
 (define (suit2number suit)
   (cond ((equal? suit 's) 4)
         ((equal? suit 'h) 3)
         ((equal? suit 'd) 2)
         ((equal? suit 'c) 1)))
```

You got up to 4 points for data abstraction
- +2 points for defining accessors (rank, suit)
- +2 points for commenting accessors
- -1 point for not using the defined accessors
- -1 point each for each non-meaningful accesssor names

You got up to 4 points for correctness of the `card-greater?` procedure:

-2 points if you switched the order of checking but other than that the program would have worked- i.e., doing something like:

```
((equal? (rank card1) (rank card2))
 (> (suit-value (suit card1)) (suit-value (suit card2))))
```

-2 points if outranks? was called incorrectly, like (outranks? card1 card2)

-1 point if you switched the importance of the suits, like heart>spade>club>diamond.

- up to -2 points if there were syntax errors (missing parenthesis), unbound variable errors, depending on severity of the error.