COMP 4958: Lab 2

Submit a zip file named `lab2.zip` containing your two source files. Maximum score: 13

1. In a file named `card_server.ex`, implement, using recursion, a registered server in a module named `CardServer` to simulate dealing a deck of 52 playing cards. Use the module name as the registered name of the server. `CardServer` should provide the following functions to its clients:

```
start()    # start registered server with "sorted" deck of 52 cards
new()      # use a new "sorted" deck of 52 cards
shuffle()  # shuffle deck of (remaining) cards
count() => integer                # return number of remaining cards
deal(n \\ 1) => {:ok, [card]} | {:error, reason} # ask server to deal n cards
```

   A new deck of cards is sorted in the order 2♣, 2♦, 2♥, 2♠, ..., A♣, A♦, A♥, A♠. (Using sorted order facilites testing.) `start/0`, `new/0`, `shuffle/0` do not return a message to the caller. The server needs to keep track of the cards that have been dealt and not deal them again from the deck. The `count/0` function returns the number of cards remaining in the deck to the caller. For `deal(n)`, either a pair consisting of the atom `:ok` and a list of `n` cards is returned, or a pair consisting of the atom `:error` and the reason for failure (a string) is returned. The error pair is returned if there are insufficient cards left in the deck, or when `n < 0`. (Choose an appropriate message for the reason in these cases.)

2. (a) Implement an Elixir function `primes/1` so that `primes(n)` returns the list of primes less than or equal to the positive integer `n`. Use a tail-recursive helper function based on the Sieve of Eratosthenes to find the primes. Note, for example, that if we want to find the primes upto & including 1,000,000, we only need to test for factors upto & including 1000. This can be used to drastically reduce the number of recursive calls.

   (b) Consider the following set of seven 6-digit primes:

$$788999, 889997, 897899, 979889, 988979, 997889, 998897.$$

   It is easy to see that the numbers are permutations of one another.

   Similarly, the following is a set of eleven 6-digit primes that are permutations of one another:

$$788789, 788897, 798887, 878789, 878987, 887987, 888779, 889877, 897887, 898787, 988877.$$

   Using the primes returned by `primes/1` above, find the size of the largest set of 6-digit primes that are permutations of one another.

   Put the two functions in a module named `Primes` in a file named `primes.exs`. You can choose any name for the function that performs the calculation in part (b). However, when `elixir primes.exs` is run, it should display the answer to that calculation.