COMP 3958: Lab 5

Submit a file named `lab5.ml` containing your functions. As before, you are limited to the functions in the standard OCaml library. Your program must also compile within `utop`; otherwise you may not get credit for this lab. You may also be asked to explain your work. Maximum score: 13

1. In class, we saw an implementation of the Sieve of Eratosthenes using imperative programming techniques — modifying arrays & references. For this exercise, you are asked to implement a version of the sieve using functional programming techniques & recursion.

   Implement a function `primes` so that `primes n` returns the list of all primes less than or equal to `n` (where `n` is a non-negative integer). For example, `primes 20` returns the list `[2; 3; 5; 7; 11; 13; 17; 19]`.

   You need to implement an auxiliary function to sieve a list of consecutive integers based on the idea of the Sieve of Eratosthenes. This sieving function needs to be tail-recursive. To reduce the number of recursive calls, note that to test whether a number is prime, we need only test for factors upto its square root.

   Note that you may only use lists & may not use arrays in your implementation. You are not allowed to use the imperative features of OCaml in this part of the lab exercise.

2. Consider the following set of 6-digit primes:

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

   It is easy to see that the numbers are permutations of one another. The set consists of 7 elements.

   Here is another set of 6-digit primes that are permutations of one another:

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

   This set has 11 elements.

   Write an OCaml function to find the size of the largest set of 6-digit primes that are permutations of one another. Name your function `permutation_size`. Its signature is

   ```
   permutation_size :  unit -> int
   ```

   Essentially, we need to group primes that are permutations of one another together. We could use some form of a dictionary, but we have not talked about the relevant modules yet. Since we've just seen the imperative features of OCaml in class, for this exercise, use an array to count how many times each permutation occurs. Note that you'll need to use part 1 or the sieve from class to find the 6-digit primes.