COMP 4958: Lab 1

Put all your functions in a module named `Lab1`.

1. Implement the following 3 list functions from basics & without using comprehension:

   (a) `join(sep, list1) -> list2`: insert `sep` between each element in `list1`; has no effect on the empty list and on a singleton list. For example,

   ```
   iex> Lab1.join(0, [3, 2, 7])
   [3, 0, 2, 0, 7]
   iex> Lab1.join(0, [1])
   [1]
   iex> Lab1.join(0, [])
   []
   ```

   (b) `flatten(deep_list) -> list`: flattens the given list of nested lists; empty list elements are discarded. For example,

   ```
   iex> Lab1.flatten([1, [[:two], 3]])
   [1, :two, 3]
   iex> Lab1.flatten([[], [[], []]])
   []
   ```

   (c) `dedup(list1) -> list2`: returns a list where all consecutive duplicated elements in `list1` are collapsed into a single element. For example,

   ```
   iex> Lab1.dedup([1, 2, 3, 3, 2, 1])
   [1, 2, 3, 2, 1]
   iex> Lab1.dedup([1, 1, 2, 2.0, :three, :three])
   [1, 2, 2.0, :three]
   ```

2. Write an Elixir function `primes(n)` that returns the list of primes less than or equal to the positive integer `n`. It uses the sieve of Eratosthenes to find the primes. You will probably need a helper function that we will call `sieve`. Write a tail-recursive version of `sieve`. Also, note that, for example, if we want to find the primes upto & includeing 1,000,000, we only need to test for factors upto & including 1000. This should drastically reduce the number of recursive calls to `sieve`.

   Use `primes` to find the number of primes less than 10 million.