Scheme Lists

Recursive List Creation

1. Create the file SchemeFlipped1.scm. Inside that file, create a method listmaker. This function should take an integer argument *n* and return a list with values ranging from *n-1* to 0. If zero is entered, it should return an empty list. **Use recursion.**

   Example:

   ```
   (listmaker 5) -> ( 4 3 2 1 0 )
   (listmaker 0) -> ()
   ```

List Alternation

2. Create the file SchemeIntro2.scm using Emacs. Define a function alternate first that returns the alternate elements from a simple list, starting from the first element.

alternate-first: Returns the elements of the list in an alternate fashion from the first element onwards. For example:

```
(alternate-firsts '(2 3 4 5 6 9 11)) -> (2 4 6 11)
```

Merging Sorted Lists

3. Create the file SchemeIntro3.scm. Write a function that merges two lists that are sorted in increasing order while maintaining the sort. If the values are the same, put the value from the left list first.

   Example:

   ```
   (mergelist '(1 5 9 13 72) '(2 4 7 8 9) ) ->
   ( 1 2 4 5 7 8 9 9 13 72)
   ```

List Intersection

4. Create the file SchemeIntro4.scm. Inside that file, write the function “intersection” that returns the intersection of two simple list parameters that represent sets. There should be no duplicates in the intersected list. Order does not matter. Don’t forget to test if the list(s) are empty. We want TWO examples with TWO different results. **You are NOT allowed to use Scheme’s built in “remove-duplicates” function or any of the “member” or “memv” functions.**

`(intersection '(1 2 3 4) '(4 -1 2 5))` **=>** `'(2 4)` ; no duplicates!! (BTW, order does not matter)

`(intersection '(5 6 7 8) '(1 2 3))` **=>** `'()'`