You are going to implement three functions in Scheme. They are:

1. The function "remove-ith", which takes an integer, i, and a list, l. The list can contain elements of any type, including other lists. The function should return a new list that has all the items from l except that the ith item has been removed. Consider the first item to be at index 0, the second item to be at index 1, etc. If index i is not valid (i.e., it is either negative or greater than the length of the list), a list equivalent to l should be returned. For example, if the function is called like this: (remove-ith 2 '("hello" (1 2 3) ((5 2) "ax") (4 5) "world"))

Then the return value should be: ("hello" (1 2 3) (4 5) "world")

1. The function "get-ith", which takes an index, i, and a list, l. The function should return the item at index i in list l. If index i is invalid, an empty list should be returned. For example, if the function is called like this: (get-ith 2 '("hello" (1 2 3) ((5 2) "ax") (4 5) "world"))

Then the return value should be: ((5 2) "ax")

1. The function "combine-elements", which takes two lists, il and l. The first list, il, is a list of indexes (all integers), some of which may be invalid, and indexes are allowed to be repeated. The second list can contain elements of any type. The return value should be a new list formed by combining the elements from l at the indexes specified by il. If some of the indexes are invalid, the return value should include empty lists. For example, if the function is called like this: (combine-elements '(-4 3 1 3 0 2 4 5 100) '("hello" 2.1 3 "hello" (4 3 5)))

Then the return value should be: (() "hello" 2.1 "hello" "hello" 3 (4 3 5) () ())

A potential hint for this function is that my implementation is very short, and it uses the provided Scheme functions "apply" and "map" as well as a lambda function and the function "get-ith" specified above. You don't necessarily have to implement it the same way.

Please put the implementations of all three Scheme functions in a single file with extension ".scm". When you are finished, email me your code to [carl.sable@cooper.edu](mailto:carl.sable@cooper.edu). Once I grade this, I won't allow re-submissions. If you want to discuss your code with me before submitting it, please set up an appointment with me. Of course, you are also welcome to email me questions related to the assignment.