TOBB University of Economics and Technology
Department of Computer Engineering
BIL395 Programming Languages
Instructor: Dr. Osman Abul

Assignment 3

Date due: March 31, 2019

Subject: Functional Programming Languages: Lisp/Scheme.

Problem: In what follows, you will find 10 programming exercises. You are expected to solve them in **Scheme** programming language.

1. Write a function which computes the area and perimeter of a regular hexagon given its side length. The function is expected to return a tuple with two components where the first is the area and the second is the perimeter.
2. Write a function which returns the list of all Fibonacci numbers not greater than a given integer. If nonsense (e.g. negative numbers, not a number) is input, the function is expected to report an error.
3. Write a function which tests whether the majority of integers are even given a simple list of numbers.
4. Write a function taking two parameters as input and deciding whether every integer in the first list appears in the second one, i.e., the first list is a subset of the second. Both lists are expected to be simple lists, otherwise you need to raise an error/warning.
5. Write a function taking two simple integer lists as input and returning a list which is the intersection of the two input lists. You may assume that the lists are already sets.
6. Write a function which removes the last element from an integer list and returns the resulting list in the reverse order.
7. Write a function which takes a simple list of numbers and sorts it in non-increasing order.

8. Write a function which outputs the maximum depth in a general integer list.
9. Write a function which decides on the structural equality of two given general lists. If two lists are structurally equal, they must have the same internal representation but the atoms in the lists are allowed to differ.
10. Write a function which accepts a simple integer list and returns the same list except all negative elements are removed from the list.

Availability: There are many freely available Scheme interpreters/compilers. For uniformity, you are strongly advised to use **Racket** which you can download from `http://racket-lang.org/`. The website also hosts the language related documentation.

Convention: Pay attention to use the following convention to name your functions: `E1` for exercise 1, `E2` for exercise 2 and so on. You can use any name for subfunctions you define.

Delivery: Collect all of your functions in a text file and email it to the teaching assistant.