COMP 348 – Fall 2019
Assignment-III

6% of final grade Due Date: October 30, 2019 – 11:59 PM

Lisp

Objectives
Functional programming using lisp.

1. Functions
For the following questions, implement the function in lisp.
Note that the examples that are provided below each item are meant to be used as auxiliary information. Your answers must work for all inputs.

A. Write a lisp function that takes a list and an integer n, and returns the first n elements of the list, as a new list, e.g.:

```
> (take-n '(1 2 3) 2)
(1 2)
```

In case n is less than 1, it returns NIL. In case n is beyond the length, the function returns a copy of the original list.

B. Modify the above function to create a deep-copy of the elements, if applicable. e.g.:

```
> (take-n-deep '((1 2) 3) 2)
((1 2) 3)
```

C. Write a function called cut-in-half that receives a list and creates a new list whose elements are the first and the second halves. e.g.:

```
> (cut-in-half '(1 2 3))
((1 2) (3))
```

In case of odd length, the first half takes the middle element.

D. Write a function called make-tree that receives a list and creates a binary tree, by calling the above cut-in-half function recursively. A node is a list that has one element. e.g.:

```
> (make-tree '(1))
(1)

> (make-tree '(1 2 3))
(((1) (2)) (3))
```

E. Write a function to calculate the height of the tree, i.e. returned by the above make-tree function. e.g.:

```
> (tree-height '())
NIL

> (tree-height '(1))
1

> (tree-height '((1) (2)))
2

> (tree-height (make-tree '(1 2 3)))
3
```

2. Small Lisp Program

Write a lisp function triangle that takes an odd number as the argument (n) and shows a triangle of printed odd numbers as shown in the following samples. If the input is even, decimal or string, it should print an appropriate message.

(triangle 7)

```
1
1 3
1 3 5
1 3 5 7
```

(triangle 9)

```
1
1 3
1 3 5
1 3 5 7
1 3 5 7 9
```

(triangle 2.5) → decimal numbers are not valid input, please enter an odd integer

(triangle 4) → even numbers are not valid input, please enter an odd integer

3. List Processing

Write a lisp function that receives a list as the input argument (the list is mixed up integers, decimals, characters and nested lists) and returns a flattened list containing all the atomic elements that are numbers, without any duplication. Sample function output is shown below:

`'(1 2 (3 1) (a 2.5) (2 4.5) ((1 2)))` → `(1 2 3 2.5 4.5)`

A. Implement the function regardless of the order of the elements in the output.

B. Implement the function with respect to the elements order. The order of the elements in the output follows the order of presence of the elements in the input list.

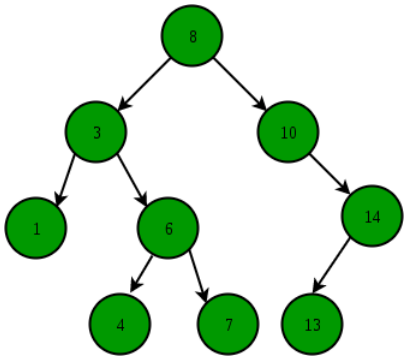
Note: Implement all functions that you use.
Hint: You may use any functions that you have previously implemented.

4. Trees

Write a lisp program to check whether a binary tree is a Binary Search Tree. A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties:

- The left sub-tree of a node has a key less than or equal to its parent node's key.
- The right sub-tree of a node has a key greater than to its parent node's key.

Example:



Example of a binary search tree

The list representing the structure of the above binary tree is as follow:

`'(8 (3 (1 () ()) (6 (4 () ())(7 () ()))) (10 () (14 (13) ())))`

5. Number and Calculations

Write a lisp function to compute series for functions *ln(1+ x)* depending. The function takes in two arguments e.g. `(compute-ln x n)`, where *x* is the input and *n* is the number of element to use in the mathematical series, as in the following:

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots + \frac{x^n}{n} \qquad -1 < x \leq 1$$

Use keyword parameter definition.

Both variables are optional with the following defaults: n = 0, n = 1.

In case of invalid inputs, the function returns NIL, as well as displaying error message to the output. Examples of invalid inputs are x or n not being a number, x outside the range, n not a positive integer, …

You are not allowed to use any inbuilt functions except predicate functions to check value type.

Deliverables

IMPORTANT: You are allowed to work on a team of 2 students at most (including yourself). You and your teammate must be in the same section. Any teams of 3 or more students will result in 0 marks for all team members. If your work on a team, ONLY one copy of the assignment is to be submitted for both members. You must make sure that you upload the assignment to the correct directory of **Assignment 3** using EAS. Assignments uploaded to the wrong directory or submitted via email will be discarded and no resubmission will be allowed.

Naming convention for uploaded file: Create one zip file, containing all needed files for your assignment using the following naming convention:
The zip file should be called *a#_studentID*, where # is the number of the assignment *studentID* is your student ID(s) number. For example, for the first assignment, student 12345678 would submit a zip file named *a1_12345678.zip*. If you work on a team and your IDs are 12345678 and 34567890, you would submit a zip file named *a1_12345678_34567890.zip*.

Submit your assignment electronically via EAS based on the instruction given by your instructor as indicated above. **Please see course outline for submission rules and format, as well as for the required demo of the assignment.** A working copy of the code and a sample output should be submitted for the tasks that require them. A text file with answers to the different tasks should be provided. Put it all in a file layout as explained below, archive it with any archiving and compressing utility, such as WinZip, WinRAR, tar, gzip, bzip2, or others. **You must keep a record of your submission confirmation.** This is your proof of submission, which you may need should a submission problem arises.

Submission Notes

Pay attention to the submission method. Assignments uploaded to the wrong system, wrong folder or category, or submitted via EMAIL will be discarded and no resubmission will be allowed.

The complete assignment must be submitted by the due date under "assignment "3" in EAS.

Grading Scheme

T#	MX	MK
1	/10	
2	/5	
3	/7	
4	/10	
5	/8	
Total:	/40	

(T# - task number, MX - max (out of), MK - your mark)

References

1. common-lisp: https://common-lisp.net/downloads