

Concordia University
Department of Computer Science and Software Engineering
COMP 348: Principles of Programming Languages
Fall 2018
Assignment 2
Evaluation: 100 pts
(5% of your final grade)
Due date and time: Before Sunday November 4, 2018 at 23:59

Note: For this assignment, you can make use of all the in-built **predicate** functions like `oddp`, `evenp`, `atom`, etc. However, you can't use any other utility functions such as `length` of list. You may need to use the "member" function in Question 3, description is available at: http://clhs.lisp.se/Body/f_mem_m.htm

Question 1 (15 pts)

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

(triangle 4)

```
1
1 2
1 2 3
1 2 3 4
```

(triangle 5)

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

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

Question 2 (15 pts)

Write a Lisp program that calculates the distance between two arbitrary points $P1:(x1 \ y1)$ and $P2:(x2 \ y2)$ based on the following formula.

$$distance(P1, P2) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

Implement your program in two different ways:

- 1- Using anonymous function (lambda)
- 2- Without applying the anonymous function.

Which method is more efficient in terms of memory allocation?

Question 3 (15 pts)

Write a lisp function that accepts a list as the input argument (the list is mixed up integers, decimals, characters and nested lists) and creates a list including all the characters in the original list without any duplication. Sample program output is shown below:

1. `'((z f) (b a 5 3.5) 6 (7) (a) c) → (z f b a c)`
2. `'((n) 2 (6 h 7.8) (w f) (n) (c) n) -> (h w f c n)`

Question 4 (10 pts)

Write a Lisp function `f-l-swap` that swaps the first and last elements of a list.

Sample Output:

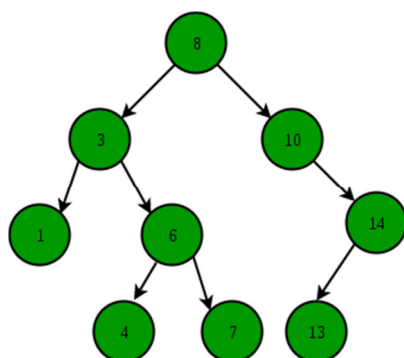
`(f-l-swap '((a d) f 10 w h)) → (h f 10 w (a d))`

`(f-l-swap '(g 6 p 10 m)) → (m 6 p 10 g)`

Question 5 (15 pts)

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 of a binary search tree

A list can be used to represent the structure of a binary tree as follow:

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

Question 6 (20 pts)

Write a lisp program to compute the series for the $\sin(x)$ and $\cos(x)$ depending on the value of n . The function takes 2 arguments (x and n) and based on the value of n calculates one of the following functions:

$$\text{sin-cos-comp}(x, n) = \begin{cases} \sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots + \frac{x^n}{n!} & -10 < x < 10, n \text{ is odd} \\ \cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots + \frac{x^n}{n!} & n \text{ is even} \end{cases}$$

You are not allowed to use any inbuilt functions except predicate functions to check value type. Your program should print an appropriate message for string and decimal values for n . Additionally, there is a limit for x when n is an odd number.

Question 7 (20 pts)

In mathematics, the **Pell numbers** are an infinite sequence of integers, that comprises the denominators of the closest rational approximation to the square root of 2. The sequence of approximations begins with

$$\frac{1}{1}, \frac{3}{2}, \frac{7}{5}, \frac{17}{12}, \frac{41}{29}, \dots$$

So the sequence of **Pell numbers** begin with 1, 2, 5, 12, 29,.... . The Pell numbers are defines as follow:

$$P_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ 2p_{n-1} + p_{n-2} & \text{otherwise} \end{cases}$$

Write a lisp program that computes the **Pell numbers** for an input argument n using:

- a) An iterative approach
- b) A recursive approach

For example `pellnumbers (6)` should return a list `(0 1 2 5 12 29 70)`

Write at least 2 test cases for each version (a and b).

Submission:

- **Assignment must be done individually (no groups are permitted).**
 - Create one zip file, containing all files for your assignment.
 - Name your zip file this way: *a1_studentID*, where *studentID* is your ID number, for the first assignment, student 123456 would submit a zip file named a1_123456.zip
- Assignments must be submitted through Moodle.

**Note: Assignment not submitted by the due date and in the correct format will not be graded
– NO EXCEPTIONS!!!!**