COMP 348: Principles of Programming Languages

Assignment 2 on Functional Programming

Summer 2020, sections AA and AB

May 30, 2020

Contents

1 General Information

Date posted: Monday May 20th, 2020.

Date due: Monday June 1st, 2020, by 23:59.

Weight: 6% of the overall grade.

2 Introduction

In this assignment you will be practicing functional programming using Lisp language.

3 Ground rules

You are allowed to work on a team of 3 students at most (including yourself). Each team should designate a leader who will submit the assignment electronically. ONLY one copy of the assignment is to be submitted.

This is an assessment exercise. You may not seek any assistance from others while expecting to receive credit. **You must work strictly within your team).** Failure to do so will result in penalties or no credit.

4 Your Assignment

Your assignment consists of five questions, each of which may be independently implemented. Some require functions and some may require complete program. Refer to each question for the details.

4.1 List Processing

For the following questions, implement the function in lisp. Some examples are provided to illustrate the behaviour of each function. Note that Your implementation must work for all form of possible inputs.

Q 1. Write a lisp function that takes a list and an integer n, and returns the last n elements of the list, as a new list, e.g.:

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

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.

Q 2. Write a function called reverse-cut-in-half that receives a list and creates a new list whose elements are the first and the second halves reversed. e.g.:

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

> (reverse-cut-in-half '((1) (2) (3) (4)))
(((3) (4)) ((1) (2)))
```

In case of odd length, the first half before reversing takes the middle element.
Show the output for `(reverse-cut-in-half '(a))`

4.2 Structures

Q 3. 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:

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

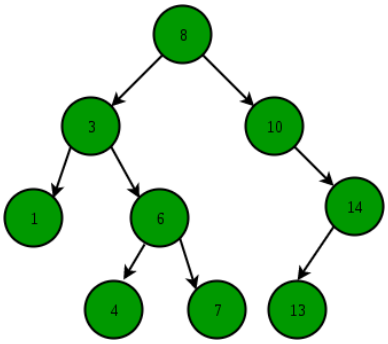
Note that the elements are given in the same order as they are received.
Partial marks are given if the order is not maintained.

Q 4. 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.

The list representing the structure of a sample binary tree is given in the following:

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



Q 5. Write a function called balancedp that takes a structure and returns true if it is balanced. A structure is balanced if number of elements and all their subelements on the left hand side is equal to the number of elements and all their subelements on the right hand side. e.g.:

```
> (balancedp '(a b c))
T

> (balancedp '(a b c d))
T

> (balancedp '(hello world (this is a test)))  ; outer list
NIL                                            ; has 3 elements

> (balancedp '(hello world (this assingment))) ; outer list
NIL                                            ; has 3 elements

> (balancedp '((1 2) (3 (1)))
T
```

4.3 Miscellaneous Programs

Q 6. Write a lisp function triangle that takes an integer as the argument and prints a triangle of stars as shown in the following figure. If the input is 0, decimal or string, it should print an appropriate error message. positive integers print left-justified triangles, whereas for the negative numbers the printed triangles are right-justified.

(triangle 7)

```
*******
******
*****
****
***
**
*
```

(triangle 4)

```
****
***
**
*
```

(triangle -5

```
*****
 ****
  ***
   **
    *
```

(triangle -1)

```
*
```

Example:

```
> (triangle 2.5)
invalid number; please enter a positive or a negative integer


> (triangle 0)
invalid number; please enter a positive or a negative integer
```

Q 7. The $3x + 1$ Conjecture: The Collatz conjecture 1931

Let T be the transformation that sends an even integer x to $x/2$ and an odd integer x to $3x + 1$. For all positive integers x, when we repeatedly apply the transformation T, we will eventually reach the integer 1.

(a) Write a function collatz that takes n as its argument and returns a list with the sequence from n to 1.

In case of invalid inputs, the function returns NIL, as well as displaying error message to the output. Examples of invalid inputs is n not being a positive number. You are not allowed to use any builtin functions except predicate functions to check value type.

```
> (collatz 4)
(2 1)

> (collatz 3)
(10 5 16 8 4 2 1)

> (collatz 10)
(5 16 8 4 2 1)

> (collatz 30)
(15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1)
```

(b) Using the above function, write a short code to print the colattz list for numbers from 1 to 20.

5 What to Submit

The assignment is to be submitted by the due date under the corresponding assignment box. Your instructor will provide you with more details.

Submission Notes

Clearly include the names and student IDs of all members of the team in the submission. Indicate the team leader.

IMPORTANT: You are allowed to work on a team of 3 students at most (including yourself). Any teams of 4 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. You must make sure that you upload the assignment to the correct assignment box on Moodle. No email submissions are accepted. Assignments uploaded to the wrong system, wrong folder, or submitted via email will be discarded and no resubmission will be allowed. Make sure you can access Moodle prior to the submission deadline. The deadline will not be extended.

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#_studids, where # is the number of the assignment, and studids is the list of student ids of all team members, separated by (_). For example, for the first assignment, student `12345678` would submit a zip file named `a1_12345678.zip`. If you work on a team of two and your IDs are `12345678` and `34567890`, you would submit a zip file named `a1_12345678_34567890.zip`. Submit your assignment electronically on Moodle based on the instruction given by your instructor as indicated above:

`https://moodle.concordia.ca`

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.

6 Grading Scheme

Q1 4 marks

Q2 5 marks

Q3 5 marks

Q4 5 marks

Q5 7 marks

Q6 7 marks

Q7 7 marks

Total: 40 marks.

References

1. Common-Lisp: `https://common-lisp.net/downloads`

2. Binary Search Tree (BST): `https://en.wikipedia.org/wiki/Binary_search_tree`

3. Collatz Conjecture: `https://en.wikipedia.org/wiki/Collatz_conjecture`