Hacettepe University Department of Computer Engineering

BBM301: Programming Languages Fall 2017

Programming Assignment

Instructors	:	Dr. Pınar Duygulu Şahin, Dr. Nazlı İkizler Cinbiş
Research Assistants	:	Cumhur Yiğit Özcan, Feyza Nur Kılıçaslan
Due Date	:	29.12.2017

Description

Aim of this experiment is to get you familiarized with functional programming. You are required to submit a single Scheme file, *<student_number>.scm*, in which you should include your solutions to all of the problems below. Each solution should simply be a function definition with the names and parameters given below. You should make sure that your program loads and works correctly with the *guile* program available on development servers of our department (located at *dev.cs.hacettepe.edu.tr*).

Please note that your Scheme implementations may also define additional, auxiliary functions if you feel they are needed. You just need to make sure that the functions indicated below are defined and they work properly. How to accomplish this task is up to you.

1. The Hyperbolic Functions (25 pts)

Define two Scheme functions, `(sinh x)` and `(cosh x)`, that can calculate corresponding hyperbolical functions for the parameter x which is an angle value in degrees. Your implementation of these functions should follow the Taylor Series representation of the hyperbolical functions given below:

$$sinh(x) = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}$$

$$cosh(x) = \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!}$$

Since these equations cannot be computed for infinite number of elements, you may stop the calculations at n=30.

```
(sinh 30)
=>0.5478534738880397
```

```
(sinh 60)
=>1.2493670505239751
```

```
(cosh 30)
=>1.1402383210764286
```

```
(cosh 60)
=>1.600286857702386
```

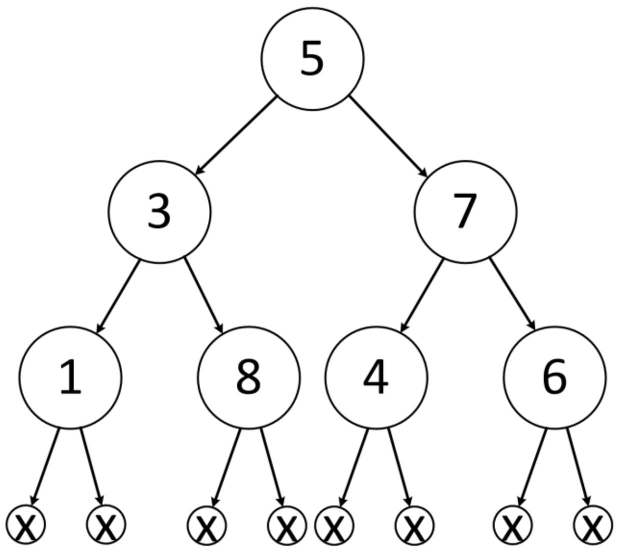
You do not have to calculate the exact numbers as given in the examples. Just make sure that you can compute the same results for at least five or six digits after the fraction part. For converting from degrees to radians, use pi=3.141592653589793238.

2. The Tree Traverse (25 pts)

For the tree given in the figure below, assume a linear list representation as follows:

```
(5 (3 (1 () ()) (8 () ())) (7 (4 () ()) (6 () ())))
```

Note that the linear form contains an empty list for each empty pointer in the leaf nodes. You are required to write three functions that are `(pre-order tree)`, `(post-order tree)` and `(in-order tree)`. These functions should return simple lists that contain the elements of the tree in corresponding order. The elements of the tree nodes can be any positive integer.



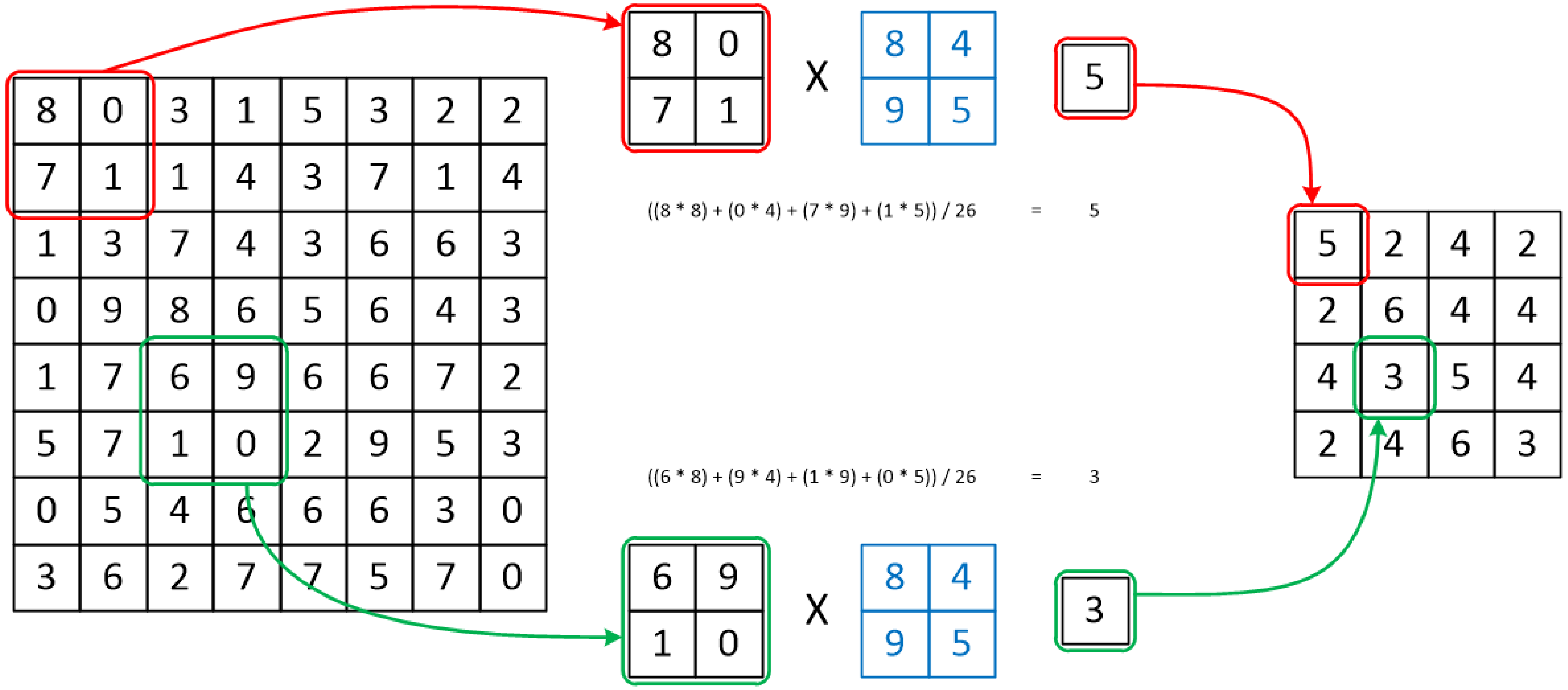
```
(pre-order '(5 (3 (1 () ()) (8 () ())) (7 (4 () ()) (6 () ()))))
=>(5 3 1 8 7 4 6)

(in-order '(5 (3 (1 () ()) (8 () ())) (7 (4 () ()) (6 () ()))))
=>(1 3 8 5 4 7 6)


(post-order '(5 (3 (1 () ()) (8 () ())) (7 (4 () ()) (6 () ()))))
=>(1 8 3 4 6 7 5)
```

3. The Matrix Convolution (50 pts)

Finally, given a source matrix (n x n) and a kernel matrix (m x m), design a function named `(convol matrix kernel)` that can create an abstract matrix (n/m x n/m). Please note that the relation between n and m can be represented by the formula n = k x m where n, k and m are positive integers. In other words, n will always be dividable by m without any remainders. The figure given below shows an example of the process of creating the abstract matrix.

In the given figure, the matrix on the left-hand side is the source matrix (of size 8 x 8). The kernel matrix (of size 2 x 2) is shown in blue. As you can see, for each of the corresponding parts of the source matrix, the values of that part are weighted using the values of the kernel matrix yielding a single value. The resulting abstract matrix is created by processing the whole source matrix.

Elements of each matrices can be any integer number and the sizes (n and m) can be any positive integer number as long as n is dividable by m without any remainders.


((8 * 8) + (0 * 4) + (7 * 9) + (1 * 5)) / 26 = 5
((6 * 8) + (9 * 4) + (1 * 9) + (0 * 5)) / 26 = 3
X
X

```
(convol '(
            (8 0 3 1 5 3 2 2)
            (7 1 1 4 3 7 1 4)
            (1 3 7 4 3 6 6 3)
            (0 9 8 6 5 6 4 3)
            (1 7 6 9 6 6 7 2)
            (5 7 1 0 2 9 5 3)
            (0 5 4 6 6 6 3 0)
            (3 6 2 7 7 5 7 0)
      )
      '(
            (8 4)
            (9 5)
      )
)
=>((5 2 4 2) (2 6 4 4) (4 3 5 4) (2 4 6 3))
```

Submission

You will submit your Scheme file named *<student_number>.scm* via the submit system. **Your file should successfully load with the following command in *guile*:**

```
(load “<student_number>.scm”)
```

Once loaded, the above example functions should be usable and behave correctly according to the given specifications.

- **Submission Format**
    - <student_number>.zip
        - <student_number>.scm