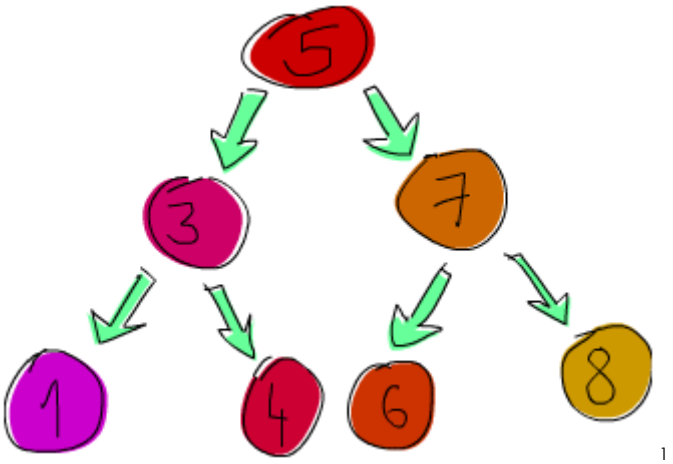
MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS

CNG242 Programming Language Concepts – Lab 3: Haskell

1. Tree Data Type

data Tree = EmptyTree | Node Integer Tree Tree deriving (Show, Eq, Ord)



[1]

- insertElement function inserts a new element to the given binary tree.
- searchElement function checks a particular element exists or not in the given binary tree.
- isEmpty function checks if a given tree is empty or not.

```
data Tree = EmptyTree | Node Integer Tree Tree deriving (Show, Eq, Ord)

insertElement x EmptyTree = Node x EmptyTree EmptyTree
insertElement x (Node a left right) = if x == a
                                      then (Node x left right)
                                      else if x < a
                                      then (Node a (insertElement x left) right)
                                      else
                                      Node a left (insertElement x right)

searchElement x EmptyTree = False
searchElement x (Node a left right) = if x == a
                                      then True
                                      else if x < a
                                      then searchElement x left
                                      else
                                      searchElement x right

isEmpty EmptyTree = True
isEmpty (Node _ _ _) = False
```

2. Lamda Abstractions

```
Prelude> let square x = x * x
Prelude> square 2
4
Prelude> (\x -> x * x) 2
4
Prelude> (\x y -> (x + y)/2) 5 7
6.0
```

[1] http://learnyouahaskell.com/making-our-own-types-and-typeclasses

3. let … in

```
myLetInFunction x = let f y z = (y + z)/2
                    in x + f 2 3
----------------------------------------
*Main> myLetInFunction 8
10.5
```

4. where

```
myWhereFunction x = x + f 2 3
                    where f y z = (y + z)/2
----------------------------------------
*Main> myWhereFunction 8
10.5
```

Practical Exercises:

1. Write a Haskell function that takes a list and a number and replicate the elements of a list a given number of times [3].

   Sample Run:
   ```
   repli "abc" 3
   "aaabbbccc"
   ```

2. Write a Haskell function that takes a list and eliminate consecutive duplicates of list elements. If a list contains repeated elements they should be replaced with a single copy of the element. The order of the elements should not be changed [3].

   Sample Run:
   ```
   compress "aaaabccaadeeee"
   "abcade"
   ```

3. The implementation of the letInFunction can be found below. You need to trace the following function and provide its output.

   ```
   letInFunction = let a = 1
                       f x = a + g x
                       g x = x + 2
                   in f 2 + let a = 4
                                g x = x +1
                                in f 3
   ```

4. Write a Haskell function which takes a list of numbers and generates a binary tree. Hint: You can use the Tree data type and insertElement function given on the first page.

References:

1. Learn You a Haskell <http://learnyouahaskell.com/chapters>
2. A Gentle Introduction to Haskell <http://www.haskell.org/tutorial/index.html>
3. H-99: Ninety-Nine Haskell Problems <https://wiki.haskell.org/H-99:_Ninety-Nine_Haskell_Problems>