|  |  |  |
| --- | --- | --- |
| American University of Sharjah  College of Engineering  Department of Computer Science & Engineering  P. O. Box 26666, Sharjah, UAE |  | **Instructors:** Dr. Michel Pasquier  **Lab Instructor:** Praveena Kolli  **Office:** EB2-12  **Phone**: 971-6-5152352  **e-mail**: pkolli@aus.edu  **Semester**: Spring 2021 |

**CMP305L - Data Structures and Algorithms Lab**

**Lab. Assignment 10 – Trees and Binary Trees**

***Objectives***

* To understand Trees and Binary Trees
* To implement recursive traversals on trees

***Instructions***

* Do not use any static or global variable. Use recursion!

***Note:***

***Lab:*** Exercises 1 and 2 (10 marks)

***Bonus*:** Exercise 3 (1 mark)

***Exercise 1:***

Write the following recursive functions that work on a general tree. The *TreeNode* structure is provided to you.

1. Depth of a node: that is the number of edges from the tree's root node to that node.
2. Height of a node: that is the number of edges on the longest downward path from that node to a leaf node.

You may use the provided main() function to continue creating nodes and the tree structure so as to test the above functions.

***Exercise 2:***

Write the following recursive functions that work on a binary tree. The*BinaryNode*structure is provided to you in appendix. The function *MakeTree* is also given, which takes an algebraic expression as string and creates a matching tree representation.

1. A function that prints the tree in *Infix* equivalent format (cf. example hereafter):

template <typename Object>

void inFix(BinaryNode<Object>\* node)

1. A function that prints the tree in *Postfix* equivalent format (cf. example hereafter):

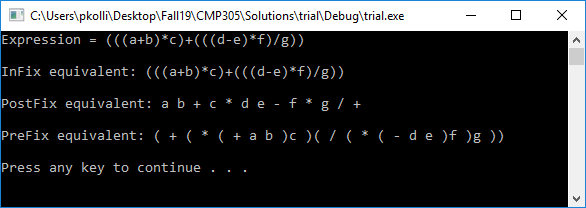
template <typename Object>

void postFix(BinaryNode<Object>\* node)

1. A Function that prints the tree in *Prefix* equivalent format (cf. example hereafter):

template <typename Object>

void preFix(BinaryNode<Object>\* node)



You should use the provided main() to test the three functions.

***Bonus:***

***Exercise 3:***

In the Array-based implementation of Binary Tree, the left child of the node at index i is at index (2\*i + 1) and the right child of the node at index (2\*i+ 2). The parent of the node at index i is at index ((i-1)/2).

Write the following functions,

1. Construct a binary tree from the given array

template <typename Object>

void ArrayToTree(BinaryNode<Object> \*&tree, Object arr[], int size, int startIndex)

1. Construct an array from the given binary tree

template <typename Object>

void treeToArray(BinaryNode<Object>\* tree, Object values[], int& index)