HCMC University of Technology
Faculty of Computer Science & Engineering

Data Structures and Algorithms

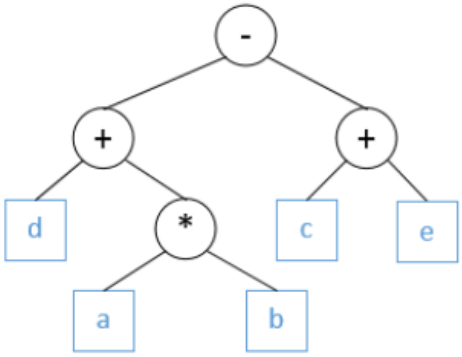
Tut 5 – Tree

Question 1

Expression **(d + (a * b)) – (c + e)** can be described by the **Expression Tree** below (LNR traverse).

Draw the Expression Trees of the following expressions:

a) (3 – a) * (b + 4)
b) a – b – c * d – e – f
c) 1 * 3 ÷ a + (b – c + d) * 7
d) (8 * 2) + (a + (b – c) * d) ÷ (5 ÷ 2)



Which **Expression Tree** among a) b) c) d) is the complete tree? Explain your answer.

Question 2

Given an empty Binary Search Tree (BST), the keys are inserted into BST one-by-one. Draw all states of the BST when inserting:

a) 15, 7, 1, 11, 9, 13, 20
b) 5, 6, 7, 8, 9
c) 100, 50, 150, 7, 55, 121, 200

Then, remove the underlined key (7) of the above trees (a, b, c). Draw the final state of the trees after removing.

Given the following data structure

```
class treeNode {
public:
    int data;
    treeNode* left = NULL;
    treeNode* right = NULL;
};
```

Question 3

a) Write a recursive function to insert a new node into the BST:

```
treeNode*recursiveInsert ( treeNode*subroot, treeNode*newNode ) {
// YOUR CODE HERE
}
```

b) Write a function to print out the path from root to the node having **searchedData**

```
void printPath ( treeNode* subroot, int searchedData ) {
// YOUR CODE HERE
}
```

c) Write a function that print out all leaves of the tree via Breadth First Traverse , LNR and NLR

```
void printLeavesBFT ( treeNode* root ) {
// YOUR CODE HERE
}
void printLeavesLNR ( treeNode* subroot ) {
// YOUR CODE HERE
}
void printLeavesNLR ( treeNode* subroot ) {
// YOUR CODE HERE
}
```

Question 4*

Propose an algorithm to check if a given Binary Tree is a BST.

algorithm *checkBST_recur (val subroot <BinaryNode>, ref min <DataType>, ref max <DataType>)*

This algorithm check if the input subroot is a BST recursively
Pre *subroot points to a root of the subtree*
Post *min and max are the smallest and largest value in the subtree*
Return *true if the subtree is a BST, false otherwise*