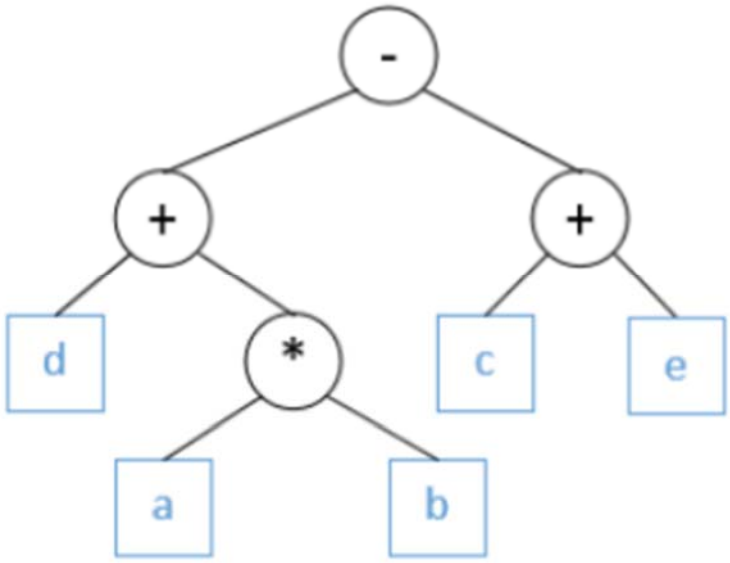
Tutorial 5 – Tree

Problem 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.

Problem 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.

Problem 3

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

```
TreeNode *loopInsert(TreeNode*  root, 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
}
```

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

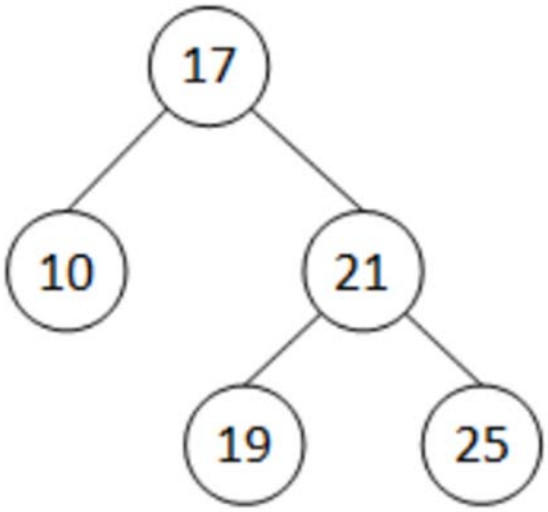
```
void printLeavesBFT(TreeNode* root) {
	// YOUR CODE HERE
}
void printLeavesLNR(TreeNode* subroot) {
	// YOUR CODE HERE
}
void printLeavesNLR(TreeNode* subroot) {
	// YOUR CODE HERE
}
```

Problem 4

Given the following array:

```
int bstTree[] = { 17, 10, 21, -1, -1, 19, 25 };
int size = 7;
```

The above array presents the BST below:



In the array:

- The negative integer means there is no node at that position.
- The first element is the root of BST.

Write a function using for/while loop to traverse and print out the tree in the order:

a) NLR

```
void  loopTraverseNLR(int* bst, int size) {
      // YOUR CODE HERE
}
```

b) LNR

```
void  loopTraverseLNR(int* bst, int size) {
      // YOUR CODE HERE
}
```

c) Breadth First

```
void  printLeavesNLR(TreeNode*  subroot) {
      // YOUR CODE HERE
}
```