Lab Assignment 6

(Week 4 – Lab A, Lab B, and Lab C)

Q1. Binary tree (BT) is a tree where each node of the tree can have maximum two children/branches, Left and Right. Besides branching, each node contains information too. While creation of BT, usually user is asked that where to insert a new node, *i.e.* left or right of a node which is already part of the BT. But in this problem, elements which are to be inserted into a BT are given in an array. These elements are two types: zero and non-zero elements. These elements are to inserted into BT in ordered manner, *i.e.* first insert into left branch of a node then insert into right branch (if these elements are non-zero). If it is zero, then that particular branch is to be left NULL for that node. Insertion will be level wise, and we will go to next level only if all branches of previous level are either inserted (due to non-zero) or left blank/NULL (due to zero). To elaborate the creation of BT, an example is presented as follows: following figure (Fig. 1) represents the BT created from the following array:

1	2	3	4	5	6	7	0	8	9	0	0	0	10	11	12	13	0	14	15	0	16	17

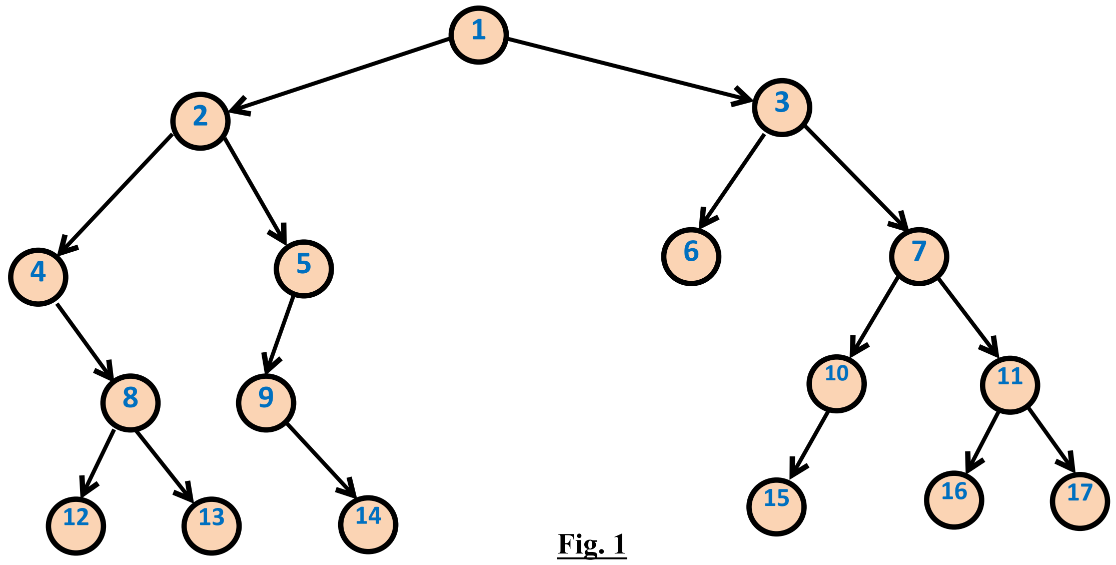


Fig. 1

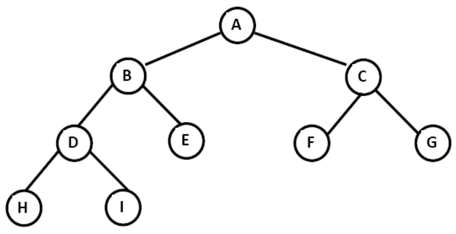
You can see here that all non-zero elements from 1 to 7 are inserted into BT as left and right child. After, node 7, level is to be changed and new element to be inserted as left child of 4, but in the array there is 0 after 7, so left child of node 4 is kept NULL and next non-zero element, i.e. 8 is inserted as the right child of node 4. Further, after insertion of node 11, level is to be changed and in new level, left and right branches of node 8 (existing left most node of previous level) are to be inserted as node 12 and node 13.

Write a program to create the BT from an inputted array (having zero and non-zero elements) and traverse the BT in following order:

(a) Pre-order (for BT of Fig. 1, it is: 1 2 4 8 12 13 5 9 14 3 6 7 10 15 11 16 17)
(b) In-order (for BT of Fig. 1, it is: 4 12 8 13 2 9 14 5 6 3 15 10 7 16 11 17)
(c) Post-order (for BT of Fig. 1, it is: 12 13 8 4 14 9 5 2 6 15 10 16 17 11 7 3 1)
(d) Level-order (for BT of Fig. 1, it is: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17)

Q2. A BT is required to be constructed as per the construction rule described in Q1. After its creation, you need to find out the path from root node to leaf node (a node is leaf node if both children are NULL) which produces maximum sum (summation of the elements retrieved from the nodes traversed between root node and leaf node). In the example of Q1, that path is 1, 3, 7, 11, 17 and maximum sum is 1+3+7+11+17 = 39. Write a program to implement the given requirement.

Q3. You have been given a list of characters as {A, B, C, D, E, F, G, H, and I). Write a C++ program to insert these elements into a complete binary tree as shown in following figure. *Do not use array to store complete binary tree, rather implement it as tree.



Also write the Recursive and Non-recursive C++ codes to traverse the binary tree as follows:

(a) Level-order: traversed node sequence should be ABCDEFGHI
(b) Spiral-order: traversed node sequence should be ABCGFEDHI

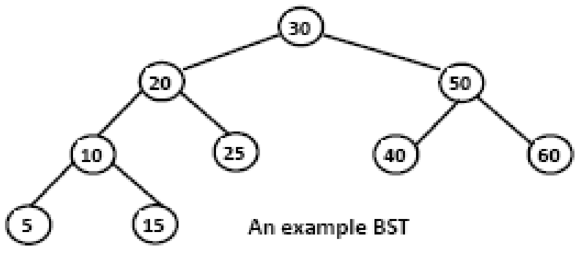
Q4. Modify the structure of the nodes of the Binary Search Tree (BST) so that the BST can contain the duplicate elements and then write a program to insert following into the modified BST:

10, 20, 30, 40, 50, 40, 35, 25, 20, 40, 18, 19, 22, 27, 30, 27

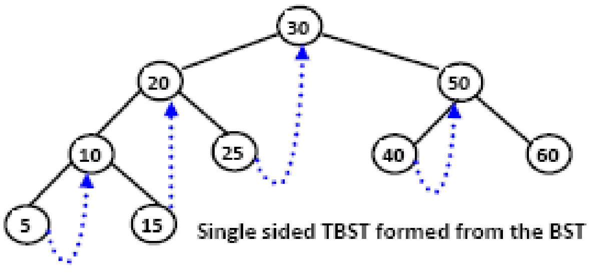
Write program (s) to perform following:

(a) Display (using In-order traversal) the elements of the modified BST
(b) Delete a user inputted element from the BST.

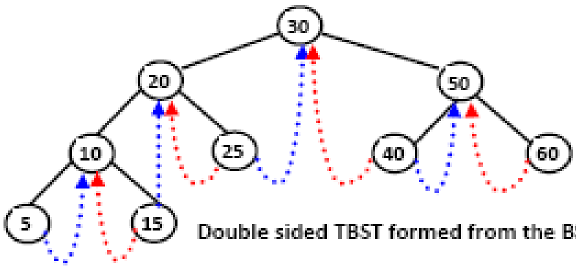
Q5. In threaded BST (TBST), the child pointers of a leaf node, N, which are NULL, are replaced (i.e. threaded) with in-order predecessor (if Left child of N is NULL) if available or in-order successor (if Right child of N is NULL) if available. Threading in TBST can be achieved in two ways, Single sided TBST, where right child pointer (which is NULL) of a leaf node, N, is replaced with the in-order successor of N; and Double Sided TBST where left child pointer (which is NULL) and right child pointer (which is NULL) of a leaf node, N, is replaced with the in-order predecessor and in-order successor of N respectively. An example of a BST, its Single sided TBST, and its Double sided TBST are shown in following figures:



An example BST



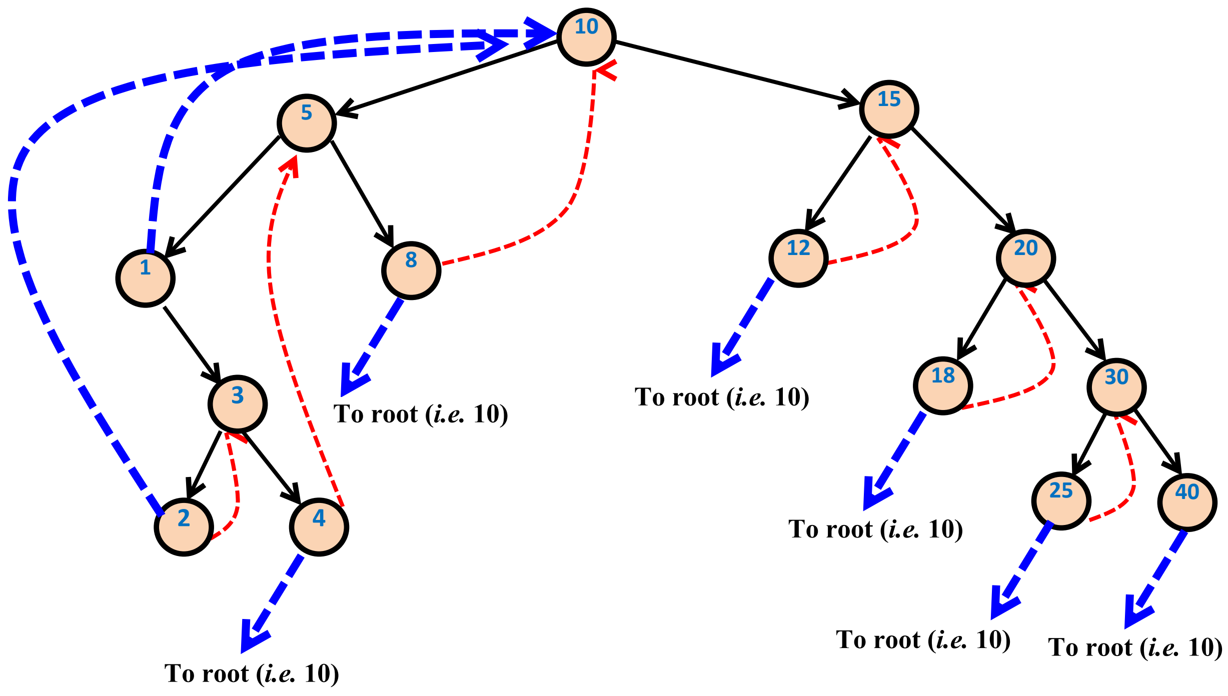
Single sided TBST formed from the BST of above example



Double sided TBST formed from the BST of above example

Write a program to create the Single sided and Double sided TBST from the given BST. After creation, write a program to delete an element/node from Single sided and Double sided TBST.

Q6. The two types of threaded BST, Single Sided TBST (SSTBST) and Double Sided TBST (DSTBST) are defined in Q5. Let's update the definition of DSTBST where instead of connecting the left pointer (if NULL) to the in-order predecessor, connect it with the root of the updated DSTBST. An example of the updated DSTBST is presented in following figure.



Write a program to create the updated DSTBST and find out the shortest path between users inputted two nodes of the updated DSTBST. For example, if user entered nodes are 1 and 12 then shortest path is 1 ➔ 10 ➔ 15 ➔ 12, or if user entered nodes are 4 and 8 then shortest path is 4 ➔ 5 ➔ 8.