DATA STRUCTURES
TUT - 8

Name-Mradul varshney
Eno-9921103137

Sol – 1 -

Pre-Order: A B F I J G K C D H L M E
Post-Order: I J K G B C L M H D E A
In-Order: I F J B K G C A D L H M E

Sol – 2 -

a) YES
b) A B C D E F G H I 0 0 0 0 0 0
c) (n-1)/2
d)2n+1 and 2(n+1)
e)efficient one as it has height of log(n)

Sol – 3 -

```
int maxDepthRecursive(TreeNode* root) {
 if(root==NULL) return 0;
 return max(maxDepthRecursive(root->left), maxDepthRecursive(root->right)) + 1;
 }
int height(TreeNode* root)
{
queue<TreeNode*> qu;
int height = 0;
int numNode = 0;
TreeNode* currNode;
if (root == NULL) {
return 0;
}
qu.push(root);
while (!qu.empty()) {
height++;
numNode = qu.size();
while (numNode--) {
currNode = qu.front();
if (currNode->left != NULL) qu.push(currNode->left);
if (currNode->right != NULL) qu.push(currNode->right);
qu.pop();
}
}
return height;
}
```

Sol – 4 -

a) In-Order: 4 2 7 5 1 3 1 0 8 6 9 1 1
Pre-order: 1 2 4 5 7 3 6 8 1 0 9 1 1
Post-order: 4 7 5 1 0 8 1 1 9 6 3 1

d)
```
findMax(TreeNode * root){
if (root == NULL) return INT_MIN;
int max = root->data;
int leftMax = findMax(root->left);
int rightMax = findMax(root->right);
if (leftMax > max) max = leftMax;
if (rightMax > max) max = rightMax;
return max;
}
int findMin(TreeNode *root){
if (root == NULL) return INT_MAX;
int max = root->data;
int leftMax = findMin(root->left);
int rightMax = findMin(root->right);
if (leftMax < max) max = leftMax;
if (rightMax < max) max = rightMax;
return max;
}
```

Sol – 5 -

```
int* postOrderIterative(struct TreeNode* root)
{
stack<TreeNode*> st;
int n = numberOfElements;
int *arr = new int[n];
int indexArr=0;
if (root == NULL) return arr;
st.push(root);
TreeNode* prev = NULL;
while (!st.empty()) {
auto currNode = st.top();
if (prev == NULL || prev->left == currNode || prev->right == currNode) {
if (currNode->left) st.push(currNode->left);
else if (currNode->right) st.push(currNode->right);
else {
st.pop();
arr[indexArr++]=currNode->val;
}
}
else if (currNode->left == prev) {
if (currNode->right) st.push(currNode->right);
else {
st.pop();
arr[indexArr++] = currNode->val;
}
}
else if (currNode->right == prev) {
st.pop();
arr[indexArr++] = currNode->val;
}
prev = currNode;
}
return arr;
}
```