# Week 6: Trees

**Lab Tutor:** For this lab-tutorial session, please discuss about the solution for each question in the lab. You may allocate about 30 minutes for each question. No need to discuss about the assignment questions and practice questions.

## Questions:

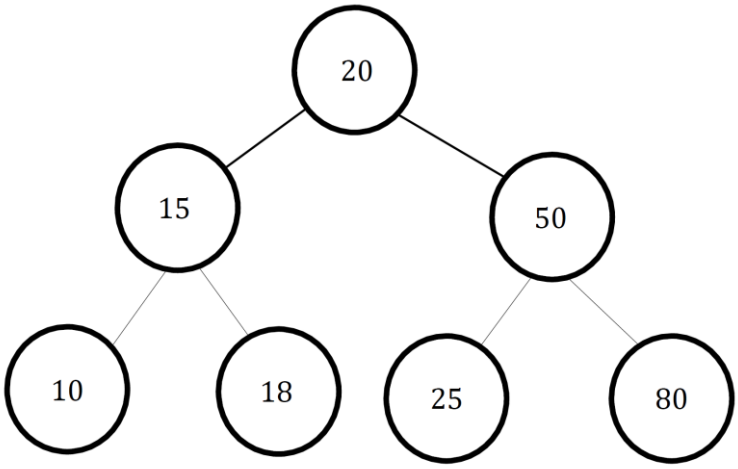
1. Write an iterative function `preOrderIterative()` that prints all items of a binary tree via pre-order traversal.

   The function prototype is given as follows:

   ```
   void preOrderIterative(BTNode *node)
   ```

   Following is the algorithm:
   1) Create a stack and push the root to the stack.
   2) Do the following steps while the stack is not empty.
      a) Peek the stack and print it
      b) Pop an item from the stack
      c) Push right child of the popped item to the stack
      d) Push left child of the popped item to the stack

   For example,

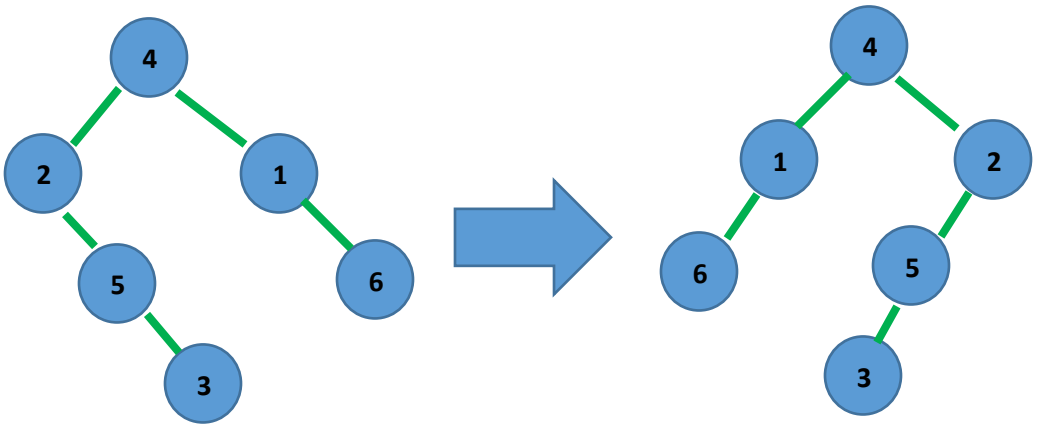
   The output is 20 15 10 18 50 25 80

2. Write a function levelOrder() that prints all items of a binary tree via breath-first traversal or level-by-level traversal.
   The function prototype is given as follows:

   ```
   void levelOrder (BTNode *node)
   ```

Using the same example in Q1, the expected output in
20 15 50 10 18 25 80

3. Write a recursive function mirrorTree() that will modify a binary tree so that the resulting tree is a mirror image of the original structure.




You should not create any intermediate or temporary trees. The function accepts a single parameter: a pointer to the root note of the binary tree to be mirrored.

```
void mirrorTree(BTNode *node);
```

A sample case is given below (the tree may be slightly different due to the random function):

```
Enter a list of numbers for a Binary Tree, terminated
by any non-digit character:
4 2 5 1 3 6 a
Original Tree
4
|---2
|       |___5
|       |       |___3
|___1
|       |___6
-------------------------------------------------
4
|---1
|       |---6
|___2
|       |---5
|       |       |---3
Mirrored Tree
```

4. Write Write a function `smallestValue()` that returns the smallest value stored in a given non-empty tree. The function accepts a pointer to the root of the given tree. You should determine the correct function prototype.

```
int smallestValue(BTNode *node);
```

Sample cases are given below:

```
Enter a list of numbers for a Binary Tree, terminated
by any non-digit character:
100001 100095 100088 a
The smallest number in the tree is 100001.

Enter a list of numbers for a Binary Tree, terminated
by any non-digit character:
70 93 1 5 -5 17 8 23 a
The smallest number in the tree is -5.
```

## Practice Questions:

1. Write a C function `printSmallerValues()` that accepts a pointer to the root node of a binary tree and prints all integers stored in the tree that are smaller than a given value m. Please use preorder tree traversal implementation. The function prototype is given as follows:

```
void printSmallerValues(BTNode *node, int m);
```

A sample case is given below (the tree may be slightly different due to the random function):

```
Enter a list of numbers for a Binary Tree, terminated
by any non-digit character:
70 31 93 14 73 94 7 23 67 99 a
Enter an integer:
50
Smaller number(s) in the tree:
31 14 7 23

Enter a list of numbers for a Binary Tree, terminated
by any non-digit character:
70 31 93 a
Enter an integer:
5
```

```
Smaller number(s) in the tree:
```

2. Write a recursive function `hasGreatGrandchild()` that prints the values stored in all nodes of a binary tree that have at least one great-grandchild. The function accepts a single parameter: a pointer to the root note of the binary tree.

```
int hasGreatGrandchild(BTNode *node);
```

*Hint: Determine the common property shared by nodes with great-grandchild nodes, and write a recursive function that computes that property.*

A sample case is given below (the tree may be slightly different due to the random function):

```
Enter a list of numbers for a Binary Tree,
terminated by any non-digit character:
70 31 93 14 73 94 7 23 67 99 25 43 56 88 77 95 a
The Binary Tree:
70
|---31
|       |---23
|       |       |---67
|       |___93
|       |       |---99
|       |       |       |---77
|       |       |       |       |___95
|       |       |       |___88
|       |       |___73
|       |       |       |---56
|___14
|       |---7
|       |       |___25
|       |       |       |___43
|       |___94
The node(s) with great grandchild:
93 31 14 70
```

3. Write a function to find the maximum depth of a binary tree.
   The function prototype is given as follows:

```
int maxDepth(BTNode *node)
```

A sample case is given below (the tree may be slightly different due to the random function):

```
Enter a list of numbers for a Binary Tree, terminated by any non-digit
character:
9 8 7 6 5 4 3 2 1 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 a
The Binary Tree:
9
|---8
|      |---2
|      |      |---1
|      |      |      |---20
|      |      |      |___18
|      |      |___17
|      |      |      |---26
|      |___7
|      |      |---10
|      |      |      |---15
|      |      |      |      |___16
|      |      |      |___14
|      |      |___5
|      |      |      |---13
|      |      |      |      |---19
|      |      |      |___25
|      |      |      |      |___27
|___6
|      |---3
|      |      |___11
|      |      |      |___12
|      |___4
|      |      |---21
|      |      |      |___24
|      |      |      |      |___28
|      |      |___22
|      |      |      |___23
The maximum depth of the binary tree is: 5
```

4. Given a binary tree and a “target” value, search the tree to see if it contains the target. The basic pattern of the searchNode() occurs in many recursive tree

algorithms: deal with the base case where the tree is empty, deal with the current node, and then use recursion to deal with the subtrees.

The function prototype is given as follows:

```
BTNode *searchNode(BTNode *root, int key);
```

A sample case is given below (the tree may be slightly different due to the random function):

```
Enter a list of numbers for a Binary Tree, terminated
by any non-digit character:
7 5 8 9 1 3 4 6 10 11 12 15 20 22 24 25 2 a
The Binary Tree:
7
|---5
|       |---6
|       |       |---10
|       |       |___2
|       |___8
|       |       |---11
|       |       |       |---24
|       |       |       |       |___25
|       |       |       |___22
|       |       |___1
|       |       |       |---20
|___9
|       |---4
|       |       |___12
|       |       |       |___15
|       |___3
Please enter a value to search:1
The node is found.
```