CE1007/CZ1007 –
DATA STRUCTURES
Assignment Questions: Binary Tree

Information:

Program templates for questions 1-4 are given as separated files (**Q1_template_BT.c, Q2_template_BT.c**, **Q3_template_BT.c**, and **Q4_template_BT.c**). You **must use** them to implement your functions. The program contains a *main()* function, which includes a switch statement to execute different functions that you should implement. You need to submit your code to the Automated Programming Submission and Assessment System (APAS) for testing. Deadline for program submission: **November 11th, 2019 (Monday) 11.59 pm**.

Assignment Questions (1-4)

1.   Write a recursive C function `identical()` to determine whether two binary trees are structurally identical, assuming the two binary trees as tree1 and tree2. This function returns 1 if two binary trees are structurally identical; otherwise, it returns 0. Note that two binary trees are structurally identical if they are both empty or if they are both non-empty and the left and the right subtrees are similar (they are made of nodes with the same values and arranged in the same way).

The function prototype is given as follows:
```
int identical(BTNode *tree1, BTNode    *tree2);
```

For example, assume that given two trees are tree1: 20 50 70 and tree2: 20 50 70. Then both trees are structurally identical.

A sample input and output session is given below:

```
1: Create a binary tree1.
2: Create a binary tree2.
3: Check whether two trees are structurally identical.
0: Quit;

Please input your choice(1/2/3/0): 1

Creating tree1:
Input an integer that you want to add to the binary tree. Any Alpha
value will be treated as NULL.
Enter an integer value for the root: 50
Enter an integer value for the Left child of 50: 20
Enter an integer value for the Right child of 50: 70
Enter an integer value for the Left child of 20: a
Enter an integer value for the Right child of 20: a
Enter an integer value for the Left child of 70: a
Enter an integer value for the Right child of 70: a
The resulting tree1 is: 20 50 70

Please input your choice(1/2/3/0): 2

Creating tree2:
Input an integer that you want to add to the binary tree. Any Alpha
value will be treated as NULL.
Enter an integer value for the root: 50
Enter an integer value for the Left child of 50: 20
Enter an integer value for the Right child of 50: 70
Enter an integer value for the Left child of 20: a
Enter an integer value for the Right child of 20: a
Enter an integer value for the Left child of 70: a
Enter an integer value for the Right child of 70: a
The resulting tree2 is: 20 50 70

Please input your choice(1/2/3/0): 3
Both trees are structurally identical.
```

```
Please input your choice(1/2/3/0): 0
```

2.   Write a C function `maxHeight()` that accepts a pointer to the root node of a binary tree and return the number of links along the longest path from the root node down to the farthest leaf node. Note that height of a given node is equal to the number of links from that node to the deepest leaf node. (Hint: **Consider that the height of an empty tree is -1**).

The function prototype is given as follows:
```
int maxHeight(BRNode *root)
```
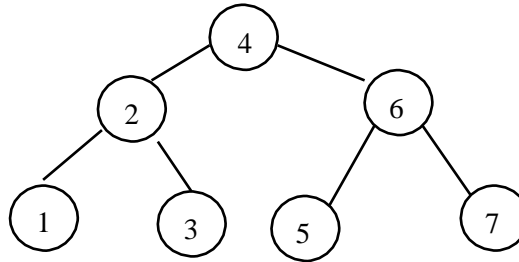

Figure 01: Binary tree

A sample input and output session is given below:

```
1: Create a binary tree.
2: Find the maximum height of the binary tree.
0: Quit;

Please input your choice(1/2/0): 1
Input an integer that you want to add to the binary tree. Any
Alpha value will be treated as NULL.

Enter an integer value for the root: 4
Enter an integer value for the Left child of 4: 2
Enter an integer value for the Right child of 4: 6
Enter an integer value for the Left child of 2: 1
Enter an integer value for the Right child of 2: 3

Enter an integer value for the Left child of 1: a
Enter an integer value for the Right child of 1: a
Enter an integer value for the Left child of 3: a
Enter an integer value for the Right child of 3: a

Enter an integer value for the Left child of 6: 5
Enter an integer value for the Right child of 6: 7

Enter an integer value for the Left child of 5: a
Enter an integer value for the Right child of 5: a
Enter an integer value for the Left child of 7: a
Enter an integer value for the Right child of 7: a

The resulting binary tree is: 1 2 3 4 5 6 7

Please input your choice(1/2/0): 2
Max height of the binary tree is: 2
Please input your choice(1/2/0): 0
```

3.   Write a C function `countOneChildNodes()` that accepts a pointer to the root node of a binary tree and return the number of nodes that have exactly one child node.

The function prototype is given as follows:
```
int countOneChildNodes(BTNode *root)
```

Some sample inputs and outputs sessions are given below:

```
1: Create a binary tree.
2: Count one child nodes in the binary tree.
0: Quit;
Please input your choice(1/2/0): 1

Input an integer that you want to add to the binary tree. Any
Alpha value will be treated as NULL.
Enter an integer value for the root: 50
Enter an integer value for the Left child of 50: 30
Enter an integer value for the Right child of 50: 70
Enter an integer value for the Left child of 30: 20
Enter an integer value for the Right child of 30: 40

Enter an integer value for the Left child of 20: a
Enter an integer value for the Right child of 20: a
Enter an integer value for the Left child of 40: 35
Enter an integer value for the Right child of 40: a
Enter an integer value for the Left child of 35: a
Enter an integer value for the Right child of 35: a

Enter an integer value for the Left child of 70: 60
Enter an integer value for the Right child of 70: a
Enter an integer value for the Left child of 60: a
Enter an integer value for the Right child of 60: a

The resulting binary tree is: 20 30 35 40 50 60 70
Please input your choice(1/2/0): 2
Number of one child nodes are: 2.

Please input your choice(1/2/0): 0
```

4.   Write a recursive C function `sumOddNodes()` that accepts a pointer to the root node of a binary tree of integers and return the sum of all odd numbers in the tree.

The function prototypes are given as follows:
```
int sumOfOddNodes(BTNode *root);
```

Some sample inputs and outputs sessions are given below:

```
1: Create a binary tree.
2: Find the sum of all odd numbers in the binary tree.
0: Quit;
Please input your choice(1/2/0): 1

Input an integer that you want to add to the binary tree. Any
Alpha value will be treated as NULL.
Enter an integer value for the root: 50
Enter an integer value for the Left child of 50: 30
Enter an integer value for the Right child of 50: 70

Enter an integer value for the Left child of 30: 25
Enter an integer value for the Right child of 30: 40
Enter an integer value for the Left child of 25: b
Enter an integer value for the Right child of 25: b
```

```
Enter an integer value for the Left child of 40: 35

Enter an integer value for the Right child of 40: b
Enter an integer value for the Left child of 35: b
Enter an integer value for the Right child of 35: b

Enter an integer value for the Left child of 70: 60
Enter an integer value for the Right child of 70: 85
Enter an integer value for the Left child of 60: a
Enter an integer value for the Right child of 60: a
Enter an integer value for the Left child of 85: a
Enter an integer value for the Right child of 85: a

The resulting binary tree is: 25 30 35 40 50 60 70 85
Please input your choice(1/2/0): 2
The sum of all odd numbers in the binary tree is: 145.

Please input your choice(1/2/0): 0
```