Lab 12 - Binary Tree Problems

Direction: Submit typed work in the Labs directory of your github repositor or dropbox, or upload to the google classroom assignment. Each part will have a template file provided named `"lab12A.cpp"` and `"lab12B.cpp"` respectively. Submit your modifications of the templates. Including any additional libraries will result in a 0.

Part A: In class

Your objective is to write the definition of the following functions

a. the function `TreeMinimum()` whose header is

```
template <typename T>
BTNode<T>* TreeMinimum(BTNode<T>* rt)
```

Given that *rt* is referencing a binary search tree, it returns the node of the tree that has the minimum *data* if the tree is not empty; otherwise, it returns NULL.

b. the function `TreeDepth()` whose header is

```
template <typename T>
int TreeDepth(BTNode<T>* rt)
```

It returns the depth of the binary tree referenced by *rt*.

Part B: Take home

Your objective is to write the definition of the following functions

a. the function `TreeSuccessor()` whose header is

```
template <typename T>
BTNode<T>* TreeSuccessor(BTNode<T>* x)
```

Given that *x* is referencing a multidirectional binary search tree, it returns the node whose data is the smallest data greater than the data of the node *x*; otherwise, it returns NULL. The node *x* is not necessarily the root of the tree.

b. the function `HasPathSum()` whose header is

```
bool HasPathSum(BTNode<int>* rt,int value)
```

Given that *rt* is referencing a binary tree, it returns true if there exists a path from *rt* to a leaf that sums to *value*; otherwise, it returns false.

c. the function `TreeSum()` whose header is

```
int TreeSum(BTNode<int>* rt)
```

It returns the sum of all the nodes of the tree. If the tree is empty, the function returns 0.