OLD Quiz 6 Version D

Due No due date **Points** 8 **Questions** 3
Available after Nov 21 at 3:30pm **Time Limit** None
Allowed Attempts Unlimited

Instructions

Quiz 6: Binary Search Trees

7 points required to pass

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	less than 1 minute	0 out of 8 *

* Some questions not yet graded

Score for this attempt: **0** out of 8 *
Submitted Dec 5 at 2:10pm
This attempt took less than 1 minute.

Question 1 — Not yet graded / 2 pts

Suppose we have this structure definition for a node in a binary search tree (BST) of **int**:

struct bt_node { int value; bt_node *left; bt_node *right };

The header node in a calling function is of type pointer-to **bt_node**. A header node of **nullptr** (or **0**) indicates an empty tree. The ordering rule for the BST is that all values less than the current node's value are to the left, and all

values greater are to the right. Duplicate values are ***not allowed***.

Write the implementation of this function that returns the number of values in a binary search tree:

size_t binary_tree_size(const bt_node *top)

{

    // this is the code you have to write

}

Your Answer:

fdsa

```
if (top == nullptr)
    return 0;
return binary_tree_size(top->left) + 1 + binary_tree_size(top->right);
```

Question 2

Not yet graded / 2 pts

Suppose we have this structure definition for a node in a binary search tree (BST) of **int**:

struct bt_node { int value; bt_node *left; bt_node *right };

The header node in a calling function is of type pointer-to **bt_node**. A header node of **nullptr** (or **0**) indicates an empty tree. The ordering rule for the BST is that all values less than the current node's value are to the left, and all values greater are to the right. Duplicate values are ***not allowed***.

Write the implementation of this function that multiplies each value in the binary tree by 2:

```
void binary_tree_multiply_by_2(bt_node *top)
{
    // this is the code you have to write
}
```

Your Answer:

```
if (top == nullptr)
    return
top->value *= 2;
binary_tree_multiply_by_2(top->left);
binary_tree_multiply_by_2(top->right);
```

Question 3

Not yet graded / 4 pts

Suppose we have this structure definition for a node in a binary search tree (BST) of **int**:

struct bt_node { int value; bt_node *left; bt_node *right };

The header node in a calling function is of type pointer-to **bt_node**. A header node of **nullptr** (or **0**) indicates an empty tree. The ordering rule for the BST is that all values less than the current node's value are to the left, and all values greater are to the right. Duplicate values are ***not allowed***.

Write the implementation of this function that *negates* (that is, multiplies by **-1**) every value in the tree. ***Hint:*** Remember to maintain the ordering rule. This can be done by swapping pointers around in the nodes.

```
void binary_tree_negate(bt_node *top)
{
    // this is the code you have to write
}
```

Your Answer:

```
if (top == nullptr)
    return
top->value *= -1;
binary_tree_negate(top->left);
binary_tree_negate(top->right);
// the left and right subtrees swap sides
bt_node *tmp = top->left;
top->left = top->right;
top->right = tmp;
```

Quiz Score: **0** out of 8