OLD Quiz 6 Version E

Due No due date **Points** 8 **Questions** 3
Available after Nov 28 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	3 minutes	5 out of 8

Score for this attempt: **5** out of 8
Submitted Nov 28 at 4:21pm
This attempt took 3 minutes.

Question 1 — 2 / 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 creates an empty BST and returns an appropriate header node value:

bt_node *mk_empty_binary_tree()

{

// this is the code you have to write

}

Your Answer:

return nullptr;

Question 2

3 / 3 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 ***sum*** of all the values in the BST, or **0** if the BST is empty:

int binary_tree_sum(const bt_node *top)

{

// this is the code you have to write

}

Your Answer:

```
if ( top == nullptr ) {
  return 0;
}
else {
  return top->value + binary_tree_sum(top->left) + binary_tree_sum(top->right);
}
```

```
if (top == nullptr)
    return 0;
return binary_tree_sum(top->left)
        + top->value
        + binary_tree_sum(top->right);
```

Question 3

0 / 3 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 inserts **val** into a binary search tree, unless **val** already exists in the tree:

```
void binary_tree_insert(bt_node **ptop, int val)
{
    // this is the code you have to write
}
```

Your Answer:

```
if (*ptop == nullptr)
    *ptop = new bt_node{ val, nullptr, nullptr};
else {
    if (val < (*ptop)->value)
        binary_tree_insert(&(*ptop)->left, val);
    else if (val > (*ptop)->value)
        binary_tree_insert(&(*ptop)->right, val);
}
```

Quiz Score: **5** out of 8