OLD Quiz 6 Version A

Due No due date **Points** 8 **Questions** 3
Available after Nov 14 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 2:07pm
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 / 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 **true** if **val** is contained in the tree, and **false** otherwise:

```
bool binary_tree_contains(const bt_node *top, int val)

{

    // this is the code you have to write

}
```

Your Answer:

```
if (top == nullptr)

    return false;

if (top->value == val)

    return true;

if (top->value < val)      // if val is in the tree, it must be to the right

    return binary_tree_contains(top->right, val);

else      // ... it must be to the left

    return binary_tree_contains(top->left, val);
```

Question 3

Not yet graded / 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: **0** out of 8