Problem 4: Find a pair with a given sum in BST

```
class Node:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None


def insert(root, val):
    if root is None:
        return Node(val)
    if val < root.val:
        root.left = insert(root.left, val)
    else:
        root.right = insert(root.right, val)
    return root


def in_order_traversal(root, target):
    stack_left = []
    stack_right = []
    curr_left = root
    curr_right = root

    done_left = False
    done_right = False
    val_left = None
    val_right = None

    while True:
        while not done_left:
            if curr_left is not None:
                stack_left.append(curr_left)
```

```
            curr_left = curr_left.left
        else:
            if len(stack_left) > 0:
                curr_left = stack_left.pop()
                val_left = curr_left.val
                curr_left = curr_left.right
            else:
                done_left = True

    while not done_right:
        if curr_right is not None:
            stack_right.append(curr_right)
            curr_right = curr_right.right
        else:
            if len(stack_right) > 0:
                curr_right = stack_right.pop()
                val_right = curr_right.val
                curr_right = curr_right.left
            else:
                done_right = True

    if val_left != val_right and val_left + val_right == target:
        return val_left, val_right

    if val_left >= val_right:
        return None

def find_pair(root, target):
    return in_order_traversal(root, target)
```

```
root = None
elements = [5,  8, 2, 6, 10]
for element in elements:
    root = insert(root, element)


target_sum = 9
pair = find_pair(root, target_sum)
if pair is not None:
    print(f"A pair with the sum {target_sum} is found: {pair[0]} and {pair[1]}")
else:
    print(f"No pair with the sum {target_sum} is found.")
```