CMPSC 412 – Lab-7 (25 points)
Binary Search Tree

Due date: 10/18/2022

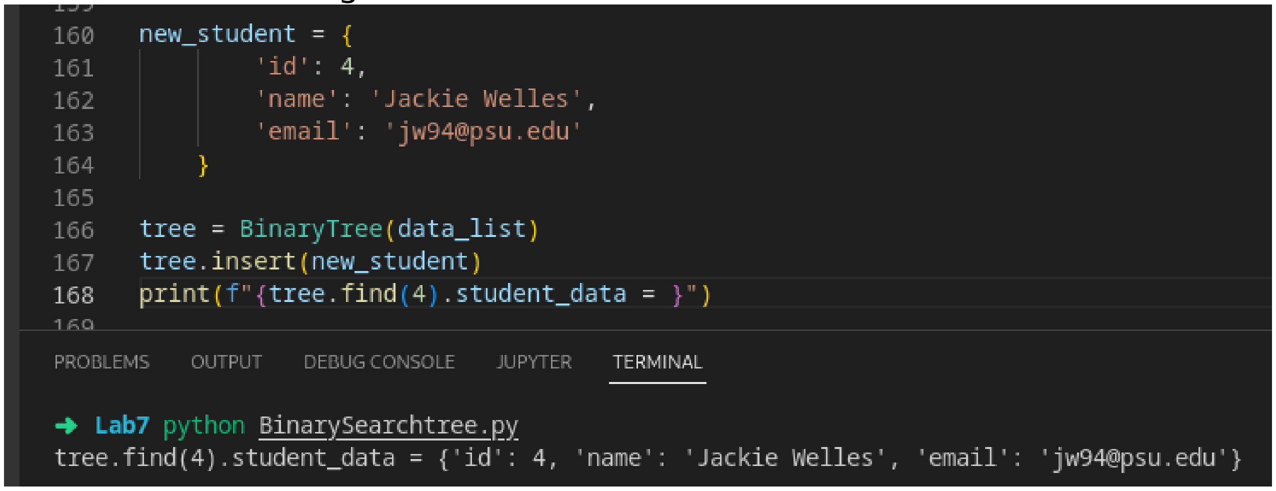
Lab Exercises:

In continuation of Lab exercise 6.1.

Exercise-1:

In continuation of Lab exercise 6.1., Develop a Binary Search Tree (BST) which can perform the following functions:

- Insert a new student with unique PSU ID and student details using BST
- Find a student using PSU ID

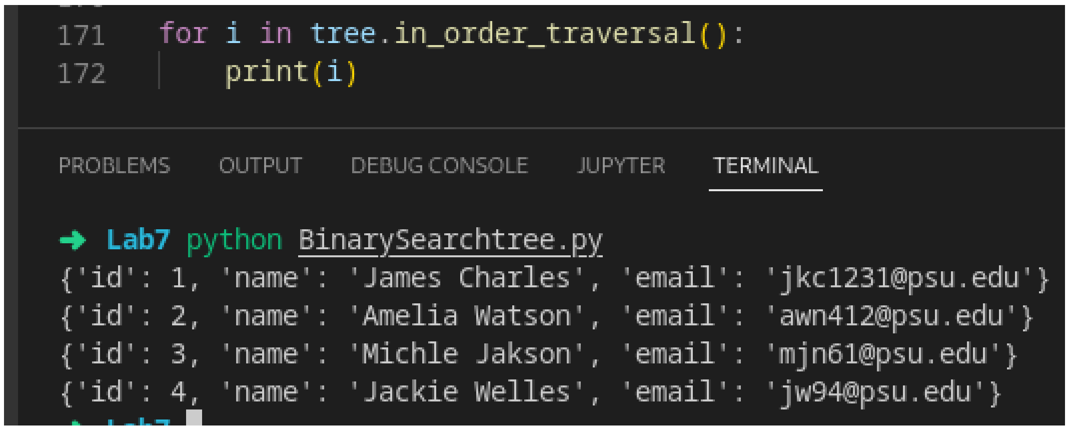


```
new_student = {
        'id': 4,
        'name': 'Jackie Welles',
        'email': 'jw94@psu.edu'
    }

tree = BinaryTree(data_list)
tree.insert(new_student)
print(f"{tree.find(4).student_data = }")
```

```
➜ Lab7 python BinarySearchtree.py
tree.find(4).student_data = {'id': 4, 'name': 'Jackie Welles', 'email': 'jw94@psu.edu'}
```

- Print each student with all details



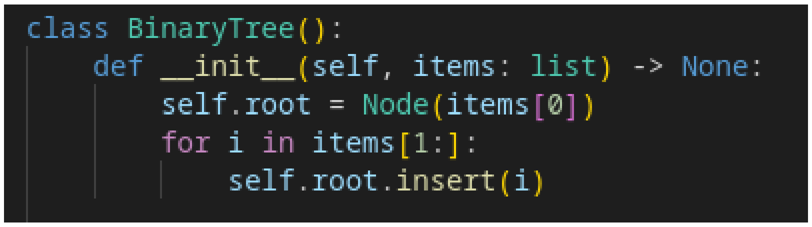
```
for i in tree.in_order_traversal():
    print(i)
```

```
➜ Lab7 python BinarySearchtree.py
{'id': 1, 'name': 'James Charles', 'email': 'jkc1231@psu.edu'}
{'id': 2, 'name': 'Amelia Watson', 'email': 'awn412@psu.edu'}
{'id': 3, 'name': 'Michle Jakson', 'email': 'mjn61@psu.edu'}
{'id': 4, 'name': 'Jackie Welles', 'email': 'jw94@psu.edu'}
```

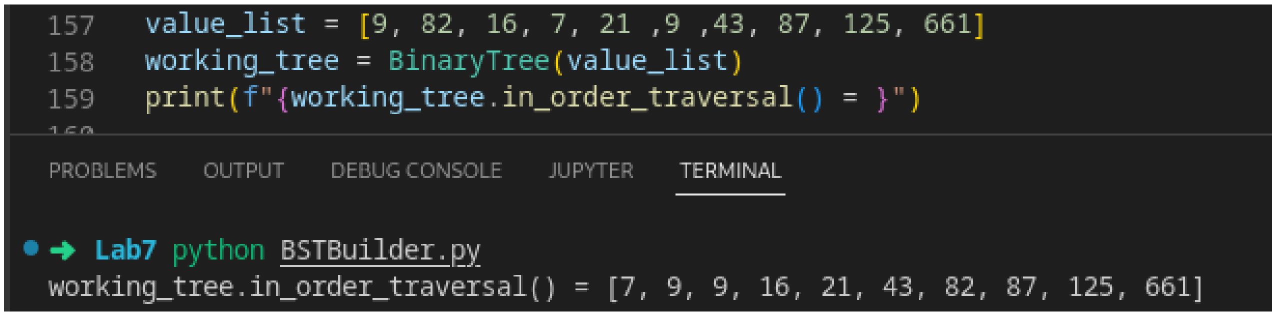
Hint: you can store the student's details as a list or dictionary as a node in the BST.

Exercise-2: (not in continuation with Exercise-1)

- Write a function which takes a list of elements and builds a Binary search tree?



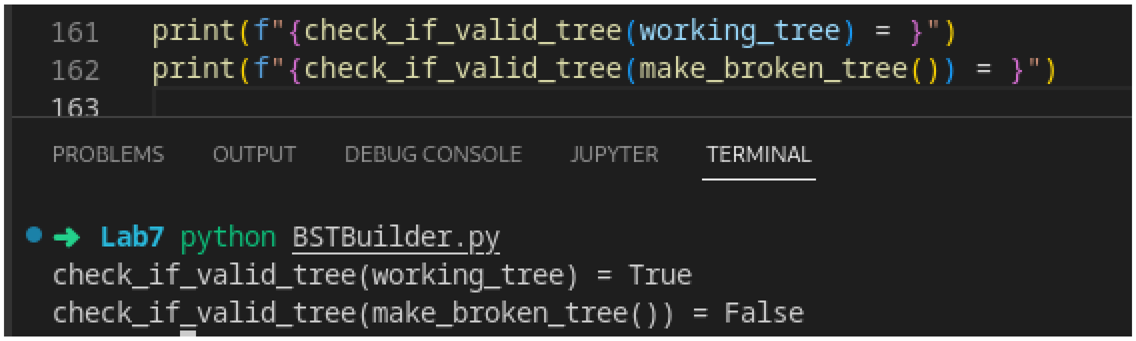
```python
class BinaryTree():
    def __init__(self, items: list) -> None:
        self.root = Node(items[0])
        for i in items[1:]:
            self.root.insert(i)
```



```python
value_list = [9, 82, 16, 7, 21 ,9 ,43, 87, 125, 661]
working_tree = BinaryTree(value_list)
print(f"{working_tree.in_order_traversal() = }")
```

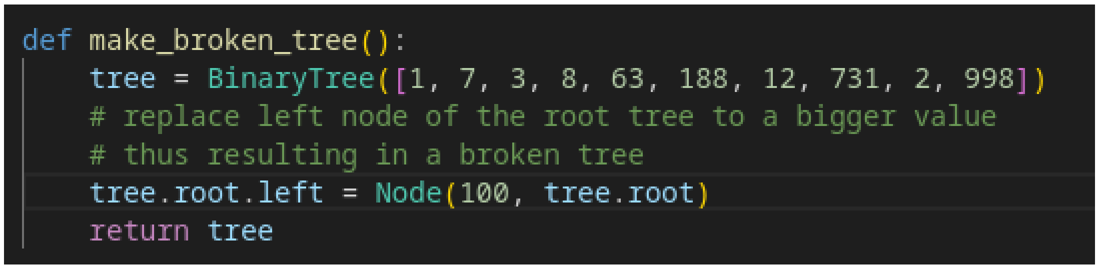
```
➜ Lab7 python BSTBuilder.py
working_tree.in_order_traversal() = [7, 9, 9, 16, 21, 43, 82, 87, 125, 661]
```

- Given a binary tree, check whether the given binary tree is a valid binary search tree (BST) or not.



```python
print(f"{check_if_valid_tree(working_tree) = }")
print(f"{check_if_valid_tree(make_broken_tree()) = }")
```

```
➜ Lab7 python BSTBuilder.py
check_if_valid_tree(working_tree) = True
check_if_valid_tree(make_broken_tree()) = False
```



```python
def make_broken_tree():
    tree = BinaryTree([1, 7, 3, 8, 63, 188, 12, 731, 2, 998])
    # replace left node of the root tree to a bigger value
    # thus resulting in a broken tree
    tree.root.left = Node(100, tree.root)
    return tree
```

Note: create a tree with minimum 10 values and demonstrate all the functions. Attach the screenshots of the results.