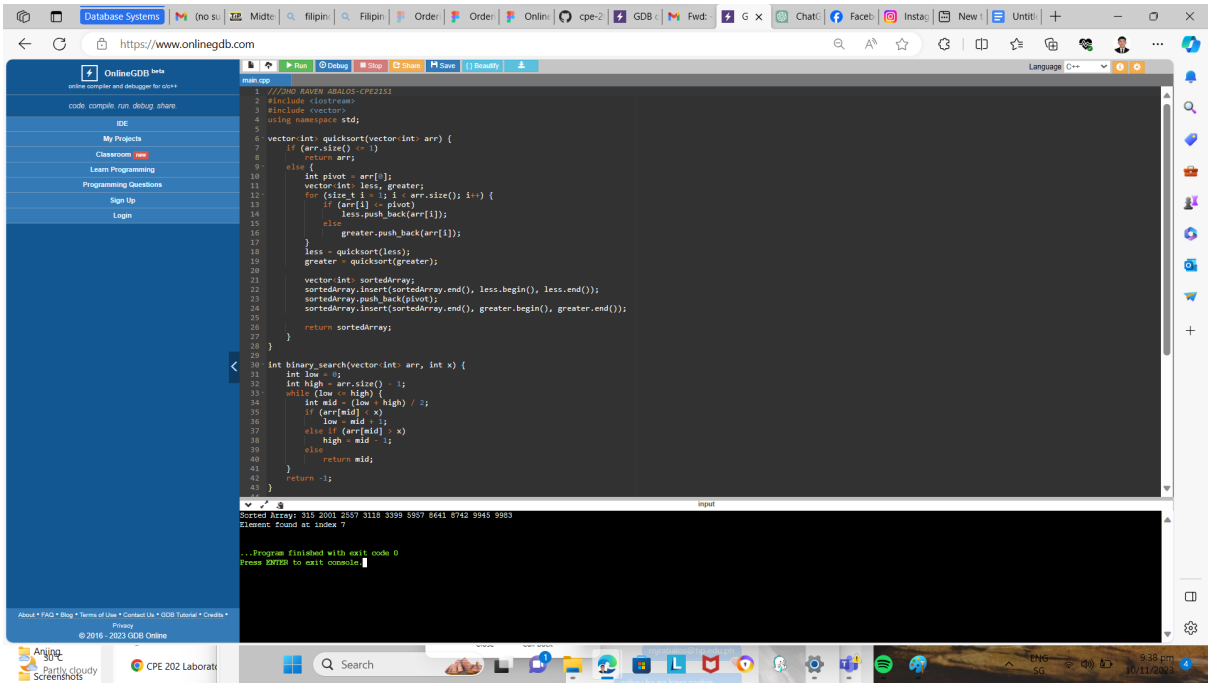
JHO RAVEN ABALOS
CPE21S1

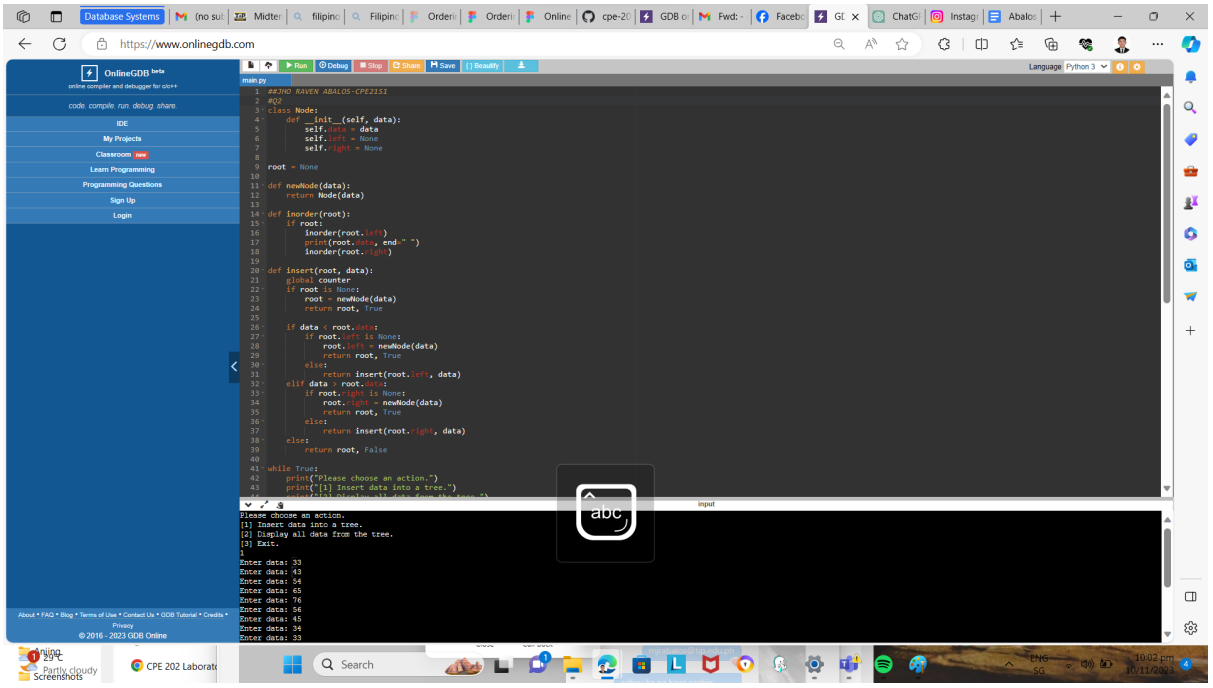
1.)




```
#include <iostream>
#include <vector>
using namespace std;

vector<int> quicksort(vector<int> arr) {
    if (arr.size() <= 1)
        return arr;
    else {
        int pivot = arr[0];
        vector<int> less, greater;
        for (size_t i = 1; i < arr.size(); i++) {
            if (arr[i] <= pivot)
                less.push_back(arr[i]);
            else
                greater.push_back(arr[i]);
        }
        less = quicksort(less);
        greater = quicksort(greater);

        vector<int> sortedArray;
        sortedArray.insert(sortedArray.end(), less.begin(), less.end());
        sortedArray.push_back(pivot);
        sortedArray.insert(sortedArray.end(), greater.begin(), greater.end());
```

```
        return sortedArray;
    }
}

int binary_search(vector<int> arr, int x) {
    int low = 0;
    int high = arr.size() - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] < x)
            low = mid + 1;
        else if (arr[mid] > x)
            high = mid - 1;
        else
            return mid;
    }
    return -1;
}

int main() {
    vector<int> unsortedArray = {9945, 3118, 3399, 2001, 9983, 8641, 2557, 8742, 5957,
315};

    vector<int> sortedArray = quicksort(unsortedArray);

    cout << "Sorted Array: ";
    for (int element : sortedArray) {
        cout << element << " ";
    }
    cout << endl;

    int searchElement = 8742;
    int result = binary_search(sortedArray, searchElement);
    if (result != -1)
        cout << "Element found at index " << result << endl;
    else
        cout << "Element not found" << endl;

    return 0;
}
```

2.)




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

root = None

def newNode(data):
    return Node(data)

def inorder(root):
    if root:
        inorder(root.left)
        print(root.data, end=" ")
        inorder(root.right)

def insert(root, data):
    global counter
    if root is None:
        root = newNode(data)
        return root, True

    if data < root.data:
        if root.left is None:
            root.left = newNode(data)
```

```
            return root, True
        else:
            return insert(root.left, data)
    elif data > root.data:
        if root.right is None:
            root.right = newNode(data)
            return root, True
        else:
            return insert(root.right, data)
    else:
        return root, False

while True:
    print("Please choose an action.")
    print("[1] Insert data into a tree.")
    print("[2] Display all data from the tree.")
    print("[3] Exit.")

    op1 = int(input())

    if op1 == 1:
        counter = 0
        while counter < 10:
            data = int(input("Enter data: "))
            if root is None:
                root = newNode(data)
            else:
                root, inserted = insert(root, data)
                if not inserted:
                    print(f"The data {data} already exists in the tree.")
            counter += 1
        else:
            print("The tree is full.")

    elif op1 == 2:
        if root is None:
            print("The tree is empty.")
        else:
            print("Nodes of the tree are: ", end="")
            inorder(root)
            print()

    elif op1 == 3:
        print("Thank you.")
        break

    else:
        print("You have entered an invalid input.")
```

```
op2 = input("\nWould you like to try again? (Y/N): ")
if op2.lower() != 'y':
    print("Thank you.")
    break
```