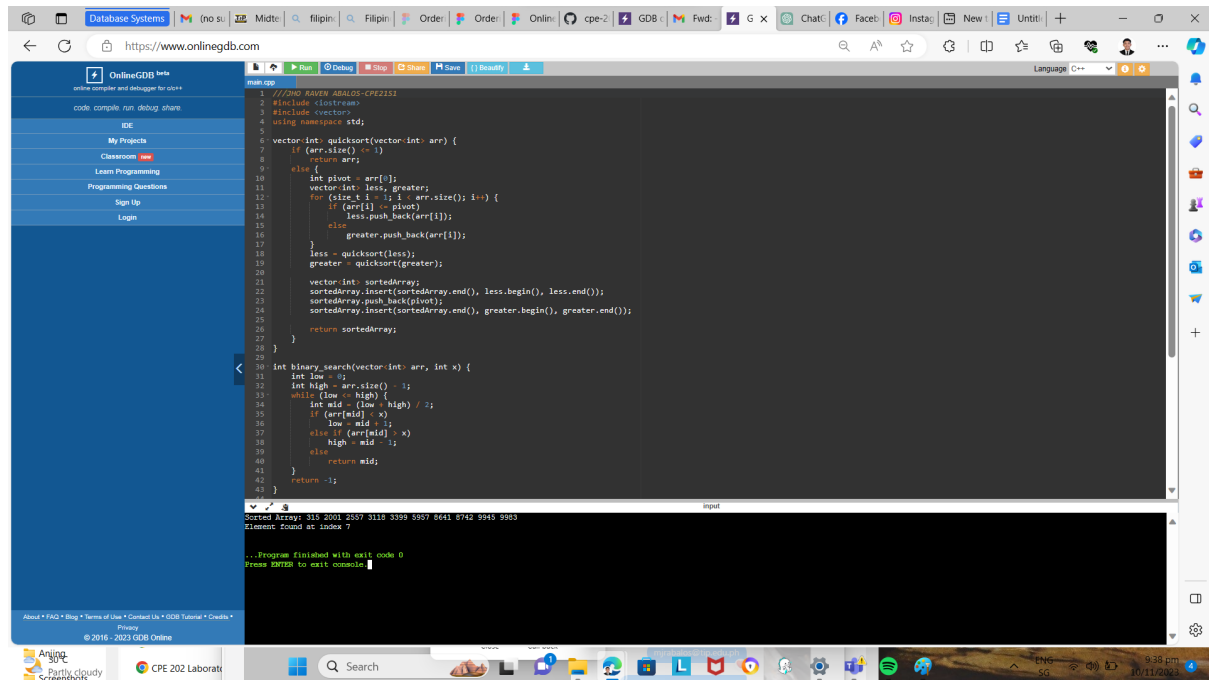


# JHO RAVEN ABALOS

## CPE21S1

1.)



The screenshot shows a web browser window with the URL <https://www.onlinegdb.com>. The page displays the OnlineGDB interface, which includes a sidebar with navigation links like 'My Projects', 'Classroom', and 'Learn Programming'. The main area shows a C++ code editor with a file named 'main.cpp'. The code implements a quicksort algorithm using vectors. Below the editor, the output window shows the sorted array: 'Sorted Array: 315 2001 2557 3115 3399 5957 5641 6742 9945 9983'. The console also shows the message '...Program finished with exit code 0' and a prompt to press ENTER to exit the console.

```
1: //JHO RAVEN ABALOS-CPE21S1
2: #include <iostream>
3: #include <vector>
4: using namespace std;
5:
6: vector<int> quicksort(vector<int> arr) {
7:     if (arr.size() <= 1)
8:         return arr;
9:     else {
10:         int pivot = arr[0];
11:         vector<int> less, greater;
12:         for (size_t i = 1; i < arr.size(); i++) {
13:             if (arr[i] <= pivot)
14:                 less.push_back(arr[i]);
15:             else
16:                 greater.push_back(arr[i]);
17:         }
18:         less = quicksort(less);
19:         greater = quicksort(greater);
20:
21:         vector<int> sortedArray;
22:         sortedArray.insert(sortedArray.end(), less.begin(), less.end());
23:         sortedArray.push_back(pivot);
24:         sortedArray.insert(sortedArray.end(), greater.begin(), greater.end());
25:
26:         return sortedArray;
27:     }
28: }
29:
30: int binary_search(vector<int> arr, int x) {
31:     int low = 0;
32:     int high = arr.size() - 1;
33:     while (low <= high) {
34:         int mid = (low + high) / 2;
35:         if (arr[mid] == x)
36:             low = mid + 1;
37:         else if (arr[mid] < x)
38:             high = mid - 1;
39:         else
40:             return mid;
41:     }
42:     return -1;
43: }
```

Sorted Array: 315 2001 2557 3115 3399 5957 5641 6742 9945 9983  
Element found at index 7  
...Program finished with exit code 0  
Press ENTER to exit console.

```
#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);

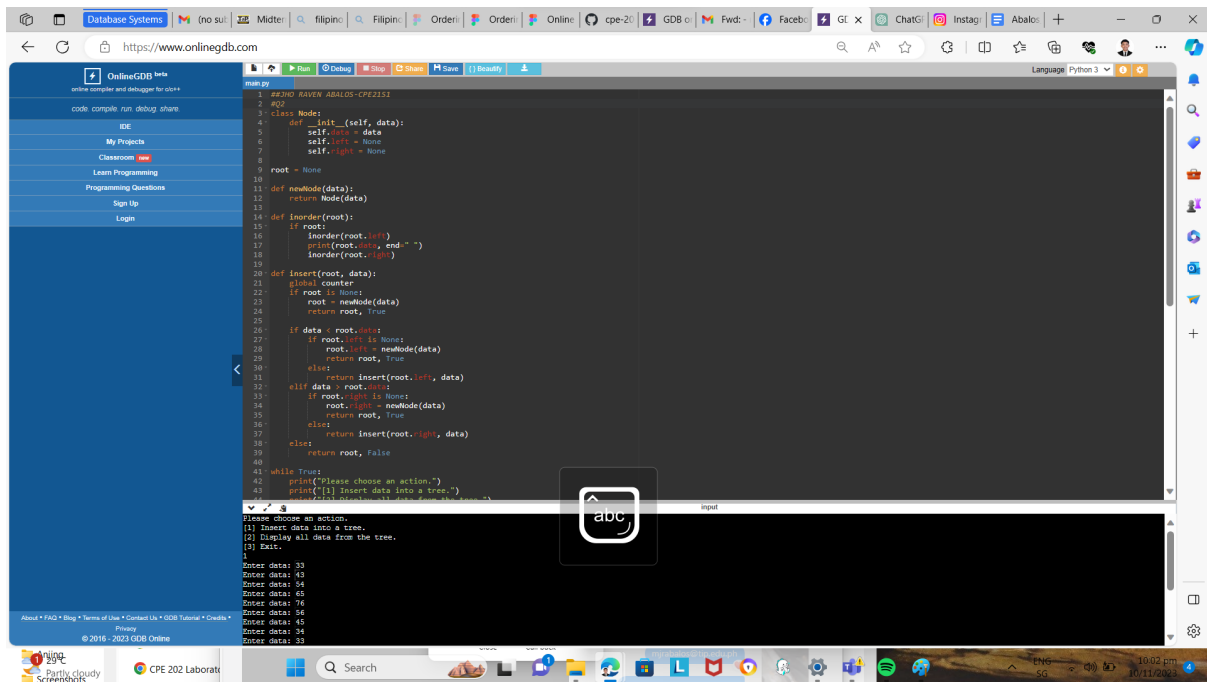
    cout << "Sorted Array: ";
    for (int element : sortedArray) {
        cout << element << " ";
    }
    cout << endl;

    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.)



```
1 #3HO RAVEN ABALOS-CPE2121
2 #Q2
3 class Node:
4     def __init__(self, data):
5         self.data = data
6         self.left = None
7         self.right = None
8
9 root = None
10
11 def newNode(data):
12     return Node(data)
13
14 def inorder(root):
15     if root:
16         inorder(root.left)
17         print(root.data, end=" ")
18         inorder(root.right)
19
20 def insert(root, data):
21     global counter
22     if root is None:
23         root = newNode(data)
24         return root, True
25
26     if data < root.data:
27         if root.left is None:
28             root.left = newNode(data)
29             return root, True
30         else:
31             return insert(root.left, data)
32     elif data > root.data:
33         if root.right is None:
34             root.right = newNode(data)
35             return root, True
36         else:
37             return insert(root.right, data)
38     else:
39         return root, False
40
41 while True:
42     print("Please choose an action.")
43     print("[1] Insert data into a tree.")
44     print("[2] Display all data from the tree.")
45     print("[3] Exit.")
46     choice = input()
47
48     if choice == "1":
49         data = input("Enter data: ")
50         root, success = insert(root, data)
51         if success:
52             print("Data inserted successfully.")
53         else:
54             print("Data already exists in the tree.")
55     elif choice == "2":
56         inorder(root)
57         print()
58     elif choice == "3":
59         break
```

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
```