CS2023 - Data Structures and Algorithms
In-class Lab Exercise

Week 7

Name: Wijetunga W.L.N.K Index Number: 200733D

You are required to answer the below questions and submit a PDF to the submission link provided under this week lab section before end of the session time (no extensions will be provided). You can either write / type your answers, but either way your answers should be readable.

GitHub repository - https://github.com/namiwijeuom/CS2023-Data-Structures-and-Algorithms-In-class-Lab-Exercises.git

Exercise:

Modify the given program to implement a binary search tree with the following basic operations. You have to define the below functions to implement the operations.

- *insertNode*()
- *deleteNode*()
- Additionally, you have to implement *traverseInOrder*() function to travese the BST inorder.

Do not modify the main function and other utility functions. You may implement any additional utility functions as you need.

Input Format

Each line has two space-separated integers. The first integer is the operator (corresponds to the integer above), while the second integer is the operand.
-1 marks the end of the input sequence.

Constraints

1 <= operator <= 2
-10000 <= operators <= 10000

Output Format

Prints the resulting BST after performing a sequence of insert and delete operations on the BST, using in order traversal. Each number is separated by a space.

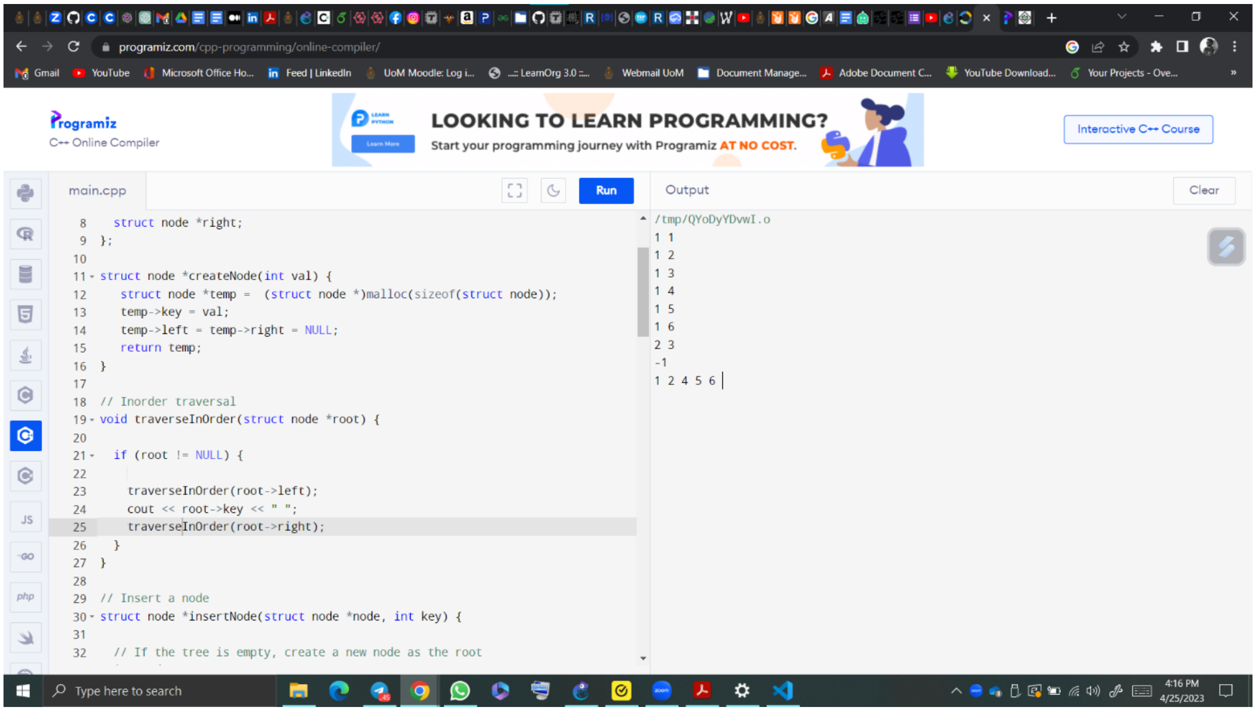
Sample Input

1 1
1 2
1 3
1 4
1 5
1 6
2 3
-1

Sample Output

1 2 4 5 6

Answer



```
  struct node *right;
};

struct node *createNode(int val) {
   struct node *temp =  (struct node *)malloc(sizeof(struct node));
   temp->key = val;
   temp->left = temp->right = NULL;
   return temp;
}

// Inorder traversal
void traverseInOrder(struct node *root) {

  if (root != NULL) {

    traverseInOrder(root->left);
    cout << root->key << " ";
    traverseInOrder(root->right);
  }
}

// Insert a node
struct node *insertNode(struct node *node, int key) {

  // If the tree is empty, create a new node as the root
```

```
/tmp/QYoDyYDvwI.o
1 1
1 2
1 3
1 4
1 5
1 6
2 3
-1
1 2 4 5 6
```