

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/namiwijeum/CS2023-Data-Structures-and-Algorithms-In-class-Lab-Exercises/tree/main/Lab%207>

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 traverse 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.

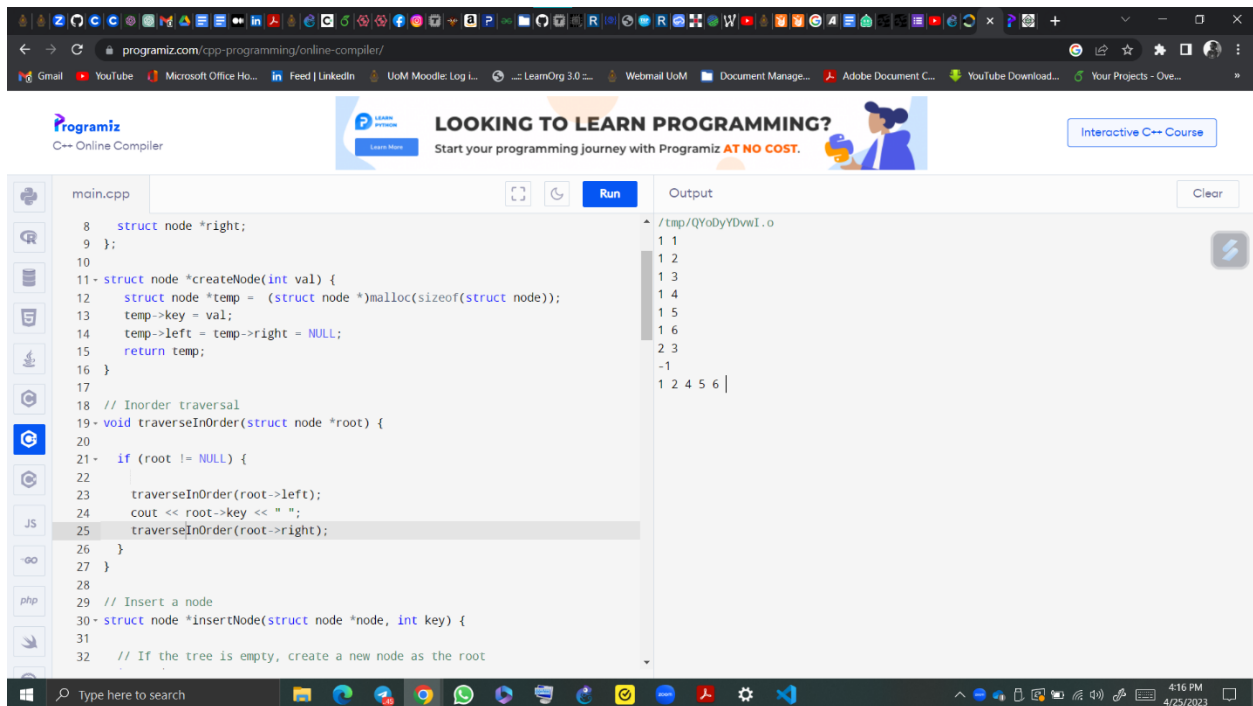
Sample Input

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

Sample Output

```
1 2 4 5 6
```

Answer



The screenshot shows a web browser with the Programiz C++ online compiler. The code in `main.cpp` implements a binary tree structure. It includes a `createNode` function to allocate and initialize a new node, and a `traverseInOrder` function to perform an inorder traversal of the tree. The `insertNode` function is also shown, which would handle the insertion of new nodes based on the input. The output of the program is displayed on the right, showing the sequence of nodes visited during the inorder traversal: 1 1, 1 2, 1 3, 1 4, 1 5, 1 6, 2 3, and -1.

```
main.cpp
8 struct node *right;
9 };
10
11 struct node *createNode(int val) {
12     struct node *temp = (struct node *)malloc(sizeof(struct node));
13     temp->key = val;
14     temp->left = temp->right = NULL;
15     return temp;
16 }
17
18 // Inorder traversal
19 void traverseInOrder(struct node *root) {
20
21     if (root != NULL) {
22         traverseInOrder(root->left);
23         cout << root->key << " ";
24         traverseInOrder(root->right);
25     }
26 }
27
28
29 // Insert a node
30 struct node *insertNode(struct node *node, int key) {
31
32     // If the tree is empty, create a new node as the root
```

Output

```
/tmp/QYoDyYDvw1.o
1 1
1 2
1 3
1 4
1 5
1 6
2 3
-1
1 2 4 5 6
```

programiz.com/cpp-programming/online-compiler/

Programiz
C++ Online Compiler

Interactive C++ Course

main.cpp

```
8 struct node *right;
9 };
10
11 struct node *createNode(int val) {
12     struct node *temp = (struct node *)malloc(sizeof(struct node));
13     temp->key = val;
14     temp->left = temp->right = NULL;
15     return temp;
16 }
17
18 // Inorder traversal
19 void traverseInOrder(struct node *root) {
20
21     if (root != NULL) {
22         traverseInOrder(root->left);
23         cout << root->key << " ";
24         traverseInOrder(root->right);
25     }
26 }
27
28
29 // Insert a node
30 struct node *insertNode(struct node *node, int key) {
31
32     // If the tree is empty, create a new node as the root
```

Output

```
/tmp/QYoDyYDvwI.o
1 1
1 2
1 4
1 6
1 8
1 5
1 6
2 5
1 7
-1
1 2 4 6 7 8
```

Type here to search

4:17 PM
4/25/2023

programiz.com/cpp-programming/online-compiler/

Programiz
C++ Online Compiler

Interactive C++ Course

main.cpp

```
8 struct node *right;
9 };
10
11 struct node *createNode(int val) {
12     struct node *temp = (struct node *)malloc(sizeof(struct node));
13     temp->key = val;
14     temp->left = temp->right = NULL;
15     return temp;
16 }
17
18 // Inorder traversal
19 void traverseInOrder(struct node *root) {
20
21     if (root != NULL) {
22         traverseInOrder(root->left);
23         cout << root->key << " ";
24         traverseInOrder(root->right);
25     }
26 }
27
28
29 // Insert a node
30 struct node *insertNode(struct node *node, int key) {
31
32     // If the tree is empty, create a new node as the root
```

Output

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

Type here to search

4:15 PM
4/25/2023