Data Structure and Algorithm(Lab)

Name: Zeeshan Ali (SU92-BSITM-F22-019)

Assignment # 03 (Lab # 11)

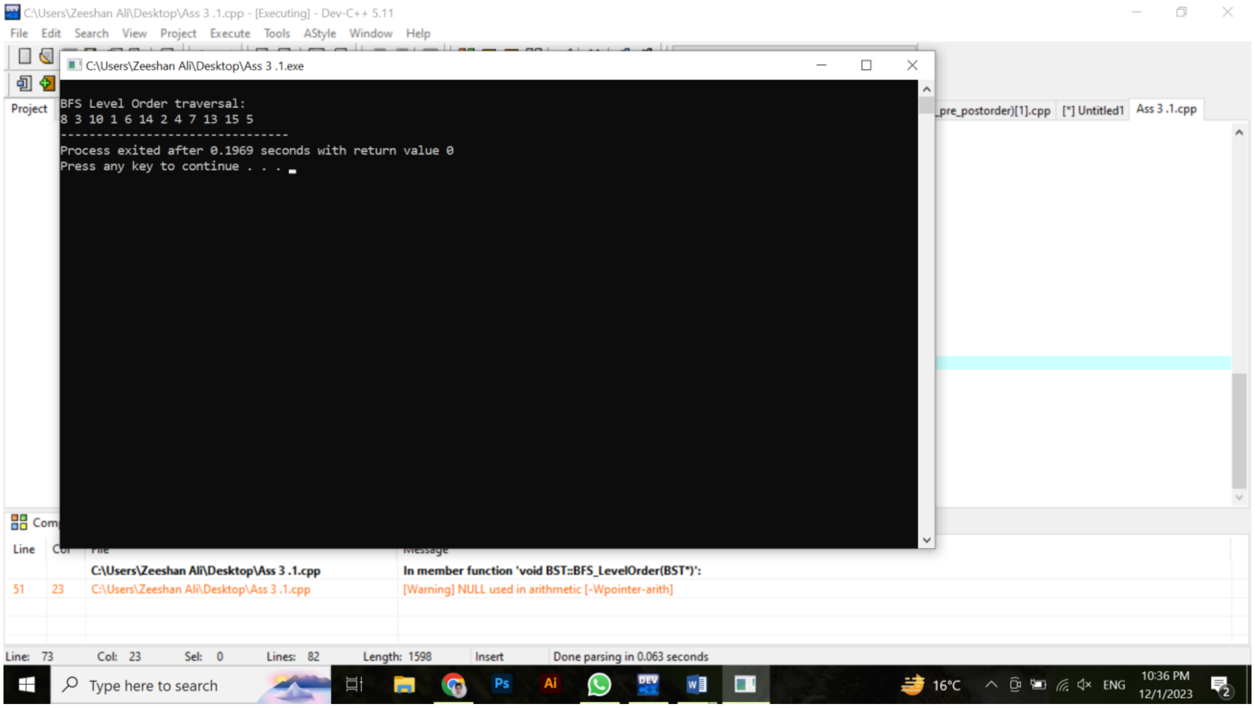
Question no 01:

Code:

```
#include <iostream>

#include <queue>

using namespace std;


class BST {

        int data;

        BST *left, *right;

public:

        BST();

        BST(int);

        BST* Insert(BST*, int);

        void Inorder(BST*);

        void preorder(BST*);

        void postorder(BST*);

        void DFS_InOrder(BST*);

        void DFS_PreOrder(BST*);

        void DFS_PostOrder(BST*);

        void BFS_LevelOrder(BST*);

};

BST ::BST(){

        data=0;

        left=NULL;

        right=NULL;
```

```
}
BST ::BST(int value)
{
        data = value;
        left = right = NULL;
}
BST* BST ::Insert(BST* root, int value)
{
        if (root==NULL) {
                return new BST(value);
        }
        if (value > root->data) {
                root->right = Insert(root->right, value);
        }
        else if (value < root->data){
                root->left = Insert(root->left, value);
        }
        return root;
}
void BST::BFS_LevelOrder(BST* root) {
    if (root == NULL) {
        return;
    }

    queue<BST*> q;
    q.push(root);

    while (q.empty()==NULL) {
        BST* current = q.front();
        cout << current->data << " ";
        if (current->left != NULL) {
            q.push(current->left);
        }
        if (current->right != NULL) {
            q.push(current->right);
        }
        q.pop();
    }
}
int main()
{
            BST b, *root = NULL;
    root = b.Insert(root, 8);
    b.Insert(root, 3);
            b.Insert(root, 10);
            b.Insert(root, 1);
            b.Insert(root, 6);
            b.Insert(root, 14);
    b.Insert(root, 4);
    b.Insert(root, 7);
    b.Insert(root, 13);
    b.Insert(root, 15);
    b.Insert(root, 2);
    b.Insert(root, 5);
    cout << "\nBFS Level Order traversal: " << endl;
    b.BFS_LevelOrder(root);
            return 0;
}
```

Output:



C:\Users\Zeeshan Ali\Desktop\Ass 3 .1.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
C:\Users\Zeeshan Ali\Desktop\Ass 3 .1.exe
Project
_pre_postorder)[1].cpp [*] Untitled1 Ass 3 .1.cpp
Com
Type here to search
16°C
ENG
10:36 PM
12/1/2023

```
BFS Level Order traversal:
8 3 10 1 6 14 2 4 7 13 15 5
--------------------------------
Process exited after 0.1969 seconds with return value 0
Press any key to continue . . .
```

Line	Col	File	Message
		C:\Users\Zeeshan Ali\Desktop\Ass 3 .1.cpp	In member function 'void BST::BFS_LevelOrder(BST*)':
51	23	C:\Users\Zeeshan Ali\Desktop\Ass 3 .1.cpp	[Warning] NULL used in arithmetic [-Wpointer-arith]

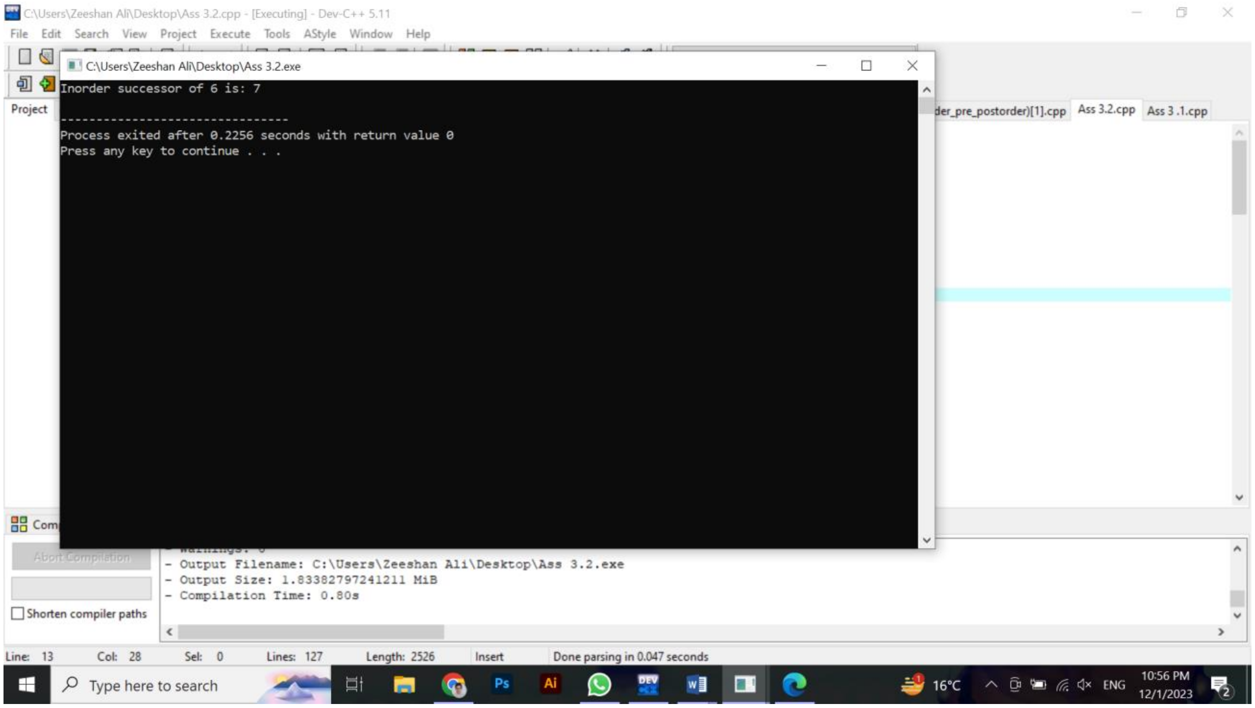
Line: 73 Col: 23 Sel: 0 Lines: 82 Length: 1598 Insert Done parsing in 0.063 seconds

Question no 02:

Code:

```
#include <iostream>a

#include <queue>

using namespace std;

class BST {

        public:

        int data;

        BST *left, *right;


public:

        BST();

        BST(int);

        void Inorder(BST*);

        void preorder(BST*);
        void postorder(BST*);
        void DFS_InOrder(BST*);
        void DFS_PreOrder(BST*);
        void DFS_PostOrder(BST*);
        void BFS_LevelOrder(BST*);
        BST* inorderSuccessor(BST*, BST*);
};
BST ::BST(){
        data=0;
        left=NULL;
        right=NULL;
}
BST ::BST(int value)
{
        data = value;
        left = right = NULL;
}
BST* BST ::Insert(BST* root, int value)
{
        if (root==NULL) {
                return new BST(value);
        }
        if (value > root->data) {

                root->right = Insert(root->right, value);
        }
        else if (value < root->data){

                root->left = Insert(root->left, value);
        }
        return root;
}
void BST ::Inorder(BST* root)
{
        if (root==NULL) {
                return;
        }
        Inorder(root->left);
        cout << root->data << endl;
        Inorder(root->right);
}
BST* BST ::inorderSuccessor(BST* root, BST* node) {
        if (node->right != NULL) {
                BST* successor = node->right;
                while (successor->left != NULL) {
                        successor = successor->left;
                }
                return successor;
        }
        BST* successor = NULL;
        while (root != NULL) {
                if (node->data < root->data) {
                        successor = root;
                        root = root->left;
                } else if (node->data > root->data) {
                        root = root->right;
                } else {
                        break;
                }
        }
        return successor;
}
int main()
{
        BST b, *root = NULL;
    root = b.Insert(root, 8);
    b.Insert(root, 3);
        b.Insert(root, 10);
        b.Insert(root, 1);
        b.Insert(root, 6);
        b.Insert(root, 14);
    b.Insert(root, 4);
    b.Insert(root, 7);
    b.Insert(root, 13);
    b.Insert(root, 15);
    b.Insert(root, 2);
    b.Insert(root, 5);
        // Assuming you have a node for which you want to find the inorder successor
        int nodeValue = 6;
        BST* nodeToFind = root;
        while (nodeToFind != NULL && nodeToFind->data != nodeValue) {
                if (nodeValue < nodeToFind->data) {
                        nodeToFind = nodeToFind->left;
                } else {
                        nodeToFind = nodeToFind->right;
                }
        }
    if (nodeToFind != NULL) {
        BST* successor = b.inorderSuccessor(root, nodeToFind);
        if (successor != NULL) {
            cout << "Inorder successor of " << nodeValue << " is: " << successor->data << std::endl;
        } else {
            cout << "No inorder successor found for " << nodeValue << std::endl;
        }
    } else {
        cout << "Node with value " << nodeValue << " not found in the BST." << std::endl;
    }
    return 0;
}
```

Output:



```
Inorder successor of 6 is: 7

--------------------------------
Process exited after 0.2256 seconds with return value 0
Press any key to continue . . .
```

```
- Output Filename: C:\Users\Zeeshan Ali\Desktop\Ass 3.2.exe
- Output Size: 1.83382797241211 MiB
- Compilation Time: 0.80s
```

C:\Users\Zeeshan Ali\Desktop\Ass 3.2.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
C:\Users\Zeeshan Ali\Desktop\Ass 3.2.exe
Project
der_pre_postorder)[1].cpp
Ass 3.2.cpp
Ass 3 .1.cpp
Abort Compilation
□ Shorten compiler paths
Line: 13 Col: 28 Sel: 0 Lines: 127 Length: 2526 Insert Done parsing in 0.047 seconds
Type here to search
16°C ENG
10:56 PM
12/1/2023

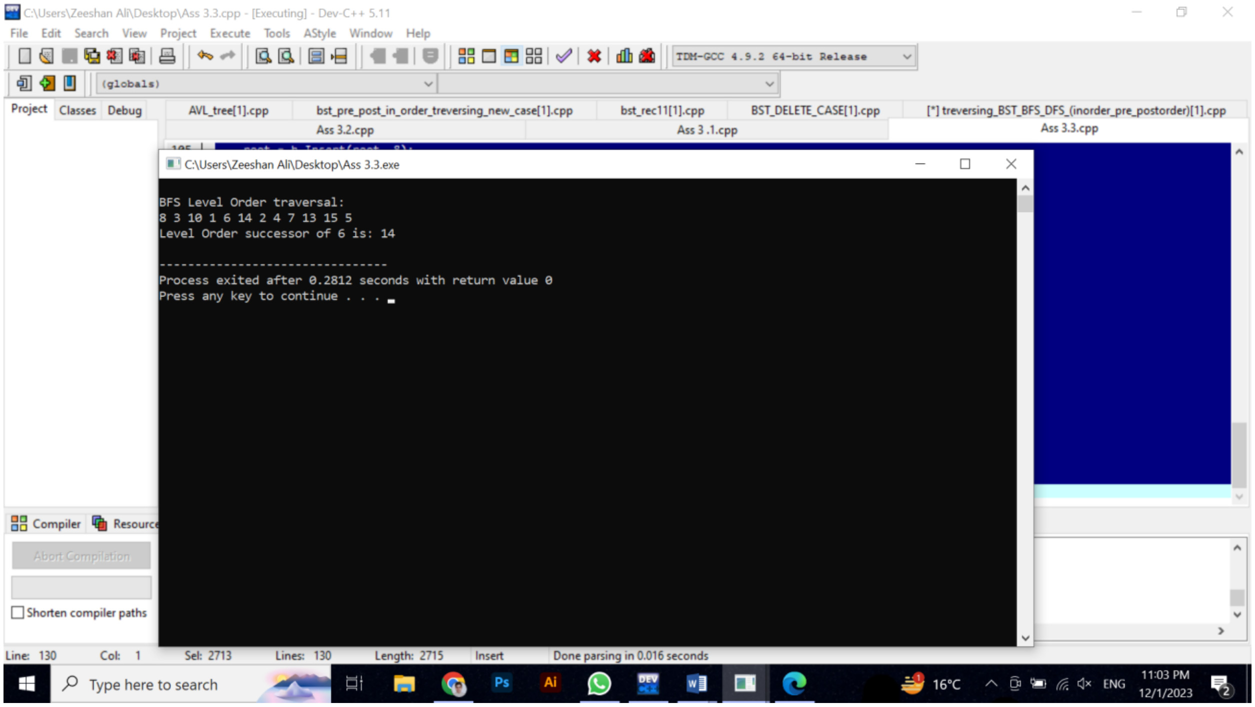
Question no 03:

Code:

```
#include <iostream>

#include <queue>

using namespace std;


class BST {

            public:

   int data;

   BST *left, *right;


public:

   BST();

   BST(int);

   BST* Insert(BST*, int);

   void Inorder(BST*);

   void preorder(BST*);

   void postorder(BST*);

   void DFS_InOrder(BST*);

   void DFS_PreOrder(BST*);

   void DFS_PostOrder(BST*);

   void BFS_LevelOrder(BST*);

   BST* findLevelOrderSuccessor(BST*, int);

};


BST::BST(){

   data=0;
```

```
    left=NULL;

    right=NULL;

}


BST::BST(int value){

    data = value;

    left = right = NULL;

}


BST* BST::Insert(BST* root, int value){

    if (root==NULL) {

        return new BST(value);

    }


    if (value > root->data) {

        root->right = Insert(root->right, value);

    }

    else if (value < root->data){

        root->left = Insert(root->left, value);

    }


    return root;

}

void BST::BFS_LevelOrder(BST* root) {

    if (root == NULL) {

        return;

    }


    queue<BST*> q;
    q.push(root);

    while (!q.empty()) {
        BST* current = q.front();
        cout << current->data << " ";
        if (current->left != NULL) {
            q.push(current->left);
        }
        if (current->right != NULL) {
            q.push(current->right);
        }
        q.pop();
    }
}

BST* BST::findLevelOrderSuccessor(BST* root, int value) {
    if (root == NULL) {
        return NULL;
    }

    queue<BST*> q;
    q.push(root);

    while (!q.empty()) {
        BST* current = q.front();
        q.pop();

        if (current->data == value) {
            if (!q.empty()) {
                return q.front();
            } else {
                return NULL;
            }
        }

        if (current->left != NULL) {
            q.push(current->left);
        }
        if (current->right != NULL) {
            q.push(current->right);
        }
    }

    return NULL;
}

int main()
{
    BST b, *root = NULL;

    root = b.Insert(root, 8);
    b.Insert(root, 3);
    b.Insert(root, 10);
    b.Insert(root, 1);
    b.Insert(root, 6);
    b.Insert(root, 14);
    b.Insert(root, 4);
    b.Insert(root, 7);
    b.Insert(root, 13);
    b.Insert(root, 15);
    b.Insert(root, 2);
    b.Insert(root, 5);
    cout << "\nBFS Level Order traversal: " << endl;
    b.BFS_LevelOrder(root);

    int nodeValue = 6;
    BST* successor = b.findLevelOrderSuccessor(root, nodeValue);
    if (successor != NULL) {
        cout << "\nLevel Order successor of " << nodeValue << " is: " << successor->data << endl;
    } else {
        cout << "\nNo Level Order successor found for " << nodeValue << endl;
    }

    return 0;
}
```

Output:



C:\Users\Zeeshan Ali\Desktop\Ass 3.3.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
AVL_tree[1].cpp
bst_pre_post_in_order_treversing_new_case[1].cpp
bst_rec11[1].cpp
BST_DELETE_CASE[1].cpp
[*] treversing_BST_BFS_DFS_(inorder_pre_postorder)[1].cpp
Ass 3.2.cpp
Ass 3 .1.cpp
Ass 3.3.cpp
C:\Users\Zeeshan Ali\Desktop\Ass 3.3.exe
Compiler
Resources
Abort Compilation
Shorten compiler paths
Line: 130 Col: 1 Sel: 2713 Lines: 130 Length: 2715 Insert Done parsing in 0.016 seconds
Type here to search
16°C
ENG
11:03 PM
12/1/2023

```
BFS Level Order traversal:
8 3 10 1 6 14 2 4 7 13 15 5
Level Order successor of 6 is: 14

--------------------------------
Process exited after 0.2812 seconds with return value 0
Press any key to continue . . .
```

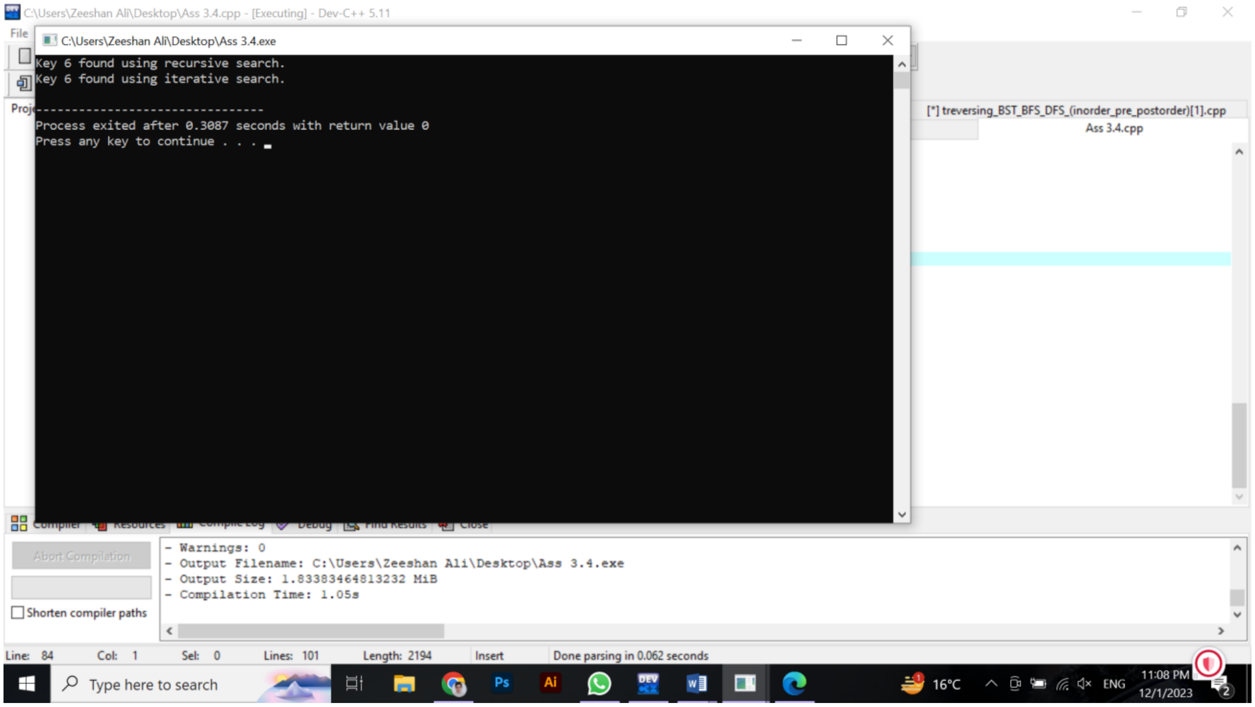
Question no 04:

Code:

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

class BST {
   int data;
   BST *left, *right;

public:
   BST();
   BST(int);
   BST* Insert(BST*, int);
   bool searchRecursive(BST*, int);
   bool searchIterative(BST*, int);
};
```

```
BST::BST(){
    data=0;
    left=NULL;
    right=NULL;
}

BST::BST(int value){
    data = value;
    left = right = NULL;
}

BST* BST::Insert(BST* root, int value){
    if (root==NULL) {
        return new BST(value);
    }

    if (value > root->data) {
        root->right = Insert(root->right, value);
    }
    else if (value < root->data){
        root->left = Insert(root->left, value);
    }

    return root;
}

bool BST::searchRecursive(BST* root, int key) {
    if (root == NULL) {
        return false;
    }

    if (root->data == key) {
        return true;
    } else if (key < root->data) {
        return searchRecursive(root->left, key);
    } else {
        return searchRecursive(root->right, key);
    }
}

bool BST::searchIterative(BST* root, int key) {
    while (root != NULL) {
        if (root->data == key) {
            return true;
        } else if (key < root->data) {
            root = root->left;
        } else {
            root = root->right;
        }
    }
    return false;
}

int main() {
    BST b, *root = NULL;

    root = b.Insert(root, 8);
    b.Insert(root, 3);
    b.Insert(root, 10);
    b.Insert(root, 1);
    b.Insert(root, 6);
    b.Insert(root, 14);
    b.Insert(root, 4);
    b.Insert(root, 7);
    b.Insert(root, 13);
    b.Insert(root, 15);
    b.Insert(root, 2);
    b.Insert(root, 5);

    int keyToSearch = 6;

    if (b.searchRecursive(root, keyToSearch)) {
        cout << "Key " << keyToSearch << " found using recursive search." << endl;
    } else {
        cout << "Key " << keyToSearch << " not found using recursive search." << endl;
    }

    if (b.searchIterative(root, keyToSearch)) {
        cout << "Key " << keyToSearch << " found using iterative search." << endl;
    } else {
        cout << "Key " << keyToSearch << " not found using iterative search." << endl;
    }

    return 0;
}
```

Output:


C:\Users\Zeeshan Ali\Desktop\Ass 3.4.cpp - [Executing] - Dev-C++ 5.11
C:\Users\Zeeshan Ali\Desktop\Ass 3.4.exe
[*] treversing_BST_BFS_DFS_(inorder_pre_postorder)[1].cpp
Ass 3.4.cpp
Compiler
Resources
Compile Log
Debug
Find Results
Close
Abort Compilation
Shorten compiler paths
Line: 84 Col: 1 Sel: 0 Lines: 101 Length: 2194 Insert Done parsing in 0.062 seconds
Type here to search
16°C
ENG
11:08 PM
12/1/2023

```
Key 6 found using recursive search.
Key 6 found using iterative search.

--------------------------------
Process exited after 0.3087 seconds with return value 0
Press any key to continue . . .
```

```
- Warnings: 0
- Output Filename: C:\Users\Zeeshan Ali\Desktop\Ass 3.4.exe
- Output Size: 1.83383464813232 MiB
- Compilation Time: 1.05s
```

Question no 05:

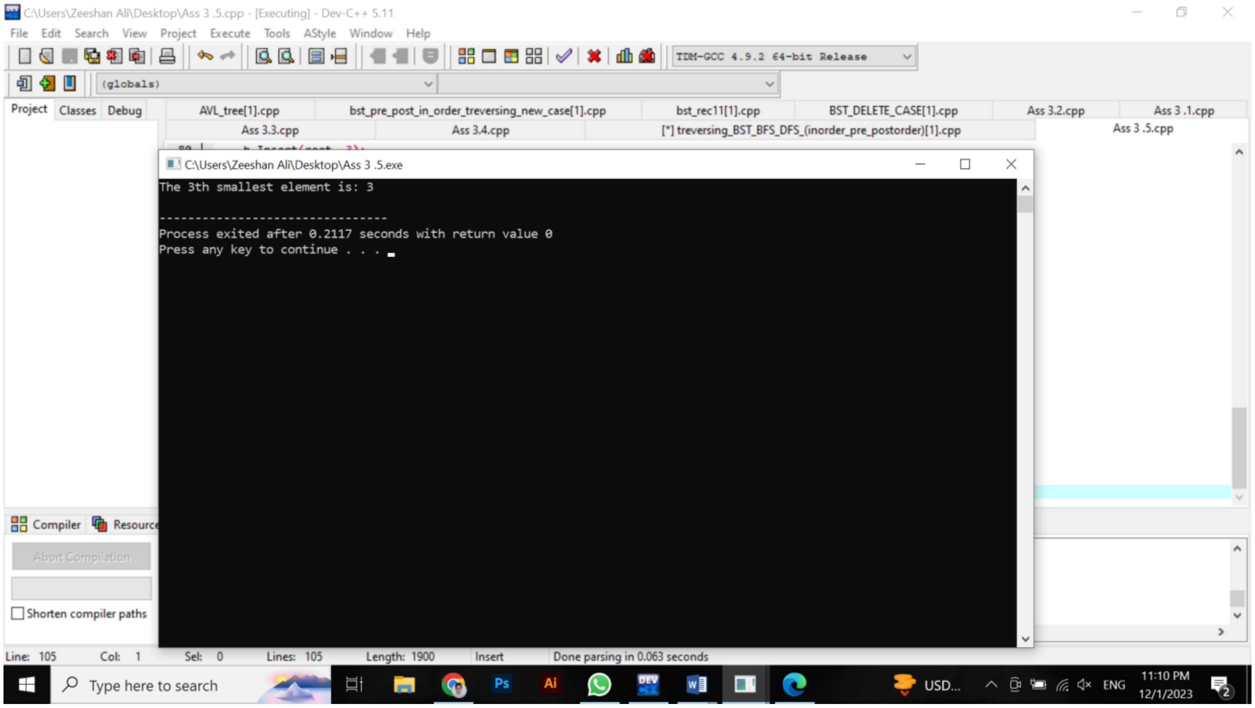
Code:

```
#include <iostream>

#include <queue>

using namespace std;


class BST {

    int data;

    BST *left, *right;


public:

    BST();

    BST(int);

    BST* Insert(BST*, int);

    void Inorder(BST*);

    int findKthSmallest(BST*, int, int&);
};

BST ::BST(){
    data=0;
    left=NULL;
    right=NULL;
}

BST ::BST(int value)
{
    data = value;
    left = right = NULL;
}

BST* BST ::Insert(BST* root, int value)
{
    if (root==NULL) {
        return new BST(value);
    }

    if (value > root->data) {
        root->right = Insert(root->right, value);
    }
    else if (value < root->data){
        root->left = Insert(root->left, value);
    }

    return root;
}
```

```
void BST ::Inorder(BST* root)
{
    if (root==NULL) {
        return;
    }
    Inorder(root->left);
    cout << root->data << " ";
    Inorder(root->right);
}

int BST::findKthSmallest(BST* root, int k, int& count) {
    if (root == NULL) {
        return -1;
    }

    int leftResult = findKthSmallest(root->left, k, count);

    if (leftResult != -1) {
        return leftResult;
    }

    count++;

    if (count == k) {
        return root->data;
    }

    return findKthSmallest(root->right, k, count);
```

```
}

int main()
{
    BST b, *root = NULL;

    root = b.Insert(root, 8);
    b.Insert(root, 3);
    b.Insert(root, 10);
    b.Insert(root, 1);
    b.Insert(root, 6);
    b.Insert(root, 14);
    b.Insert(root, 4);
    b.Insert(root, 7);
    b.Insert(root, 13);
    b.Insert(root, 15);
    b.Insert(root, 2);
    b.Insert(root, 5);

    int k = 3;

    int count = 0;
    int kthSmallest = b.findKthSmallest(root, k, count);

    if (kthSmallest != -1) {
        cout << "The " << k << "th smallest element is: " << kthSmallest << endl;
    } else {
        cout << "Invalid value of K." << endl;
    }
    return 0;

}
```

Output:



C:\Users\Zeeshan Ali\Desktop\Ass 3 .5.cpp - [Executing] - Dev-C++ 5.11

C:\Users\Zeeshan Ali\Desktop\Ass 3 .5.exe

```
The 3th smallest element is: 3

--------------------------------
Process exited after 0.2117 seconds with return value 0
Press any key to continue . . .
```

Line: 105 Col: 1 Sel: 0 Lines: 105 Length: 1900 Insert Done parsing in 0.063 seconds