**Experiment No. 6b**

**Title :** Implementation of Binary Tree Traversal.

**Problem Statement :** Write a C++ code to implement binary tree traversal with following functionalities

Insert() ,Inorder() ,Preorder() ,Postorder()

**Algorithm:**

**Step 1:** Start

**Step 2:** Create a structure binary\_tree\_node with left and right pointer of the same structure and a data variable

**Step 3:** Create a class bonary\_tree with the root as instance of structure binary\_tree\_node and point it to NULL and create a function isempty() which check for is empty of tree

**Step 4:** Create class function insert() to insert a value into the binary tree which checks for root node existence and if no new node will be root node else the data is compared with the node data and put to left or right .

**Step 5:** Create a inorder() function which traverse from left node to parent and then to right most node and this continues till root node

**Step 6:** Create a preorder() function of class which prints root node then recursively traverse left sub tree and then right sub tree.

**Step 7: C**reate a postorder() function which recursively traverse through left sub tree then recursively traverse through right sub tree., then visit the bode.

**Step 8:** Instantiate a object of class and insert values to the tree and call the above functions and observe the output.

**Step 9:** Stop

**Program :**

#include<iostream>;

using namespace std;

//binary tree node declaration

struct bintree\_node{

bintree\_node \*left;

bintree\_node \*right;

char data;

};

class bintree\_class{

bintree\_node \*root;

public:

bintree\_class()

{

root=NULL;

}

int isempty()

{

return(root==NULL);

}

void insert\_node(int item);

void inorder\_seq();

void inorder(bintree\_node \*);

void postorder\_seq();

void postorder(bintree\_node \*);

void preorder\_seq();

void preorder(bintree\_node \*);

};

void bintree\_class::insert\_node(int item)

{

bintree\_node \*p=new bintree\_node;

bintree\_node \*parent;

p->data=item;

p->left=NULL;

p->right=NULL;

parent=NULL;

if(isempty())

root=p;

else

{

bintree\_node \*ptr;

ptr=root;

while(ptr!=NULL)

{

parent=ptr;

if(item<ptr->data)

ptr=ptr->right;

else

ptr=ptr->left;

}

if(item<parent->data)

parent->left=p;

else

parent->right=p;

}

}

void bintree\_class::inorder\_seq()

{

inorder(root);

}

void bintree\_class::inorder(bintree\_node \*ptr)

{

if(ptr!=NULL)

{

inorder(ptr->left);

cout<<" "<<ptr->data<<" ";

inorder(ptr->right);

}

}

void bintree\_class::postorder\_seq()

{

postorder(root);

}

void bintree\_class::postorder(bintree\_node \*ptr)

{

if(ptr!=NULL)

{

postorder(ptr->left);

postorder(ptr->right);

cout<<" "<<ptr->data<<" ";

}

}

void bintree\_class::preorder\_seq()

{

preorder(root);

}

void bintree\_class::preorder(bintree\_node \*ptr)

{

if(ptr!=NULL)

{

cout<<" "<<ptr->data<<" ";

preorder(ptr->left);

preorder(ptr->right);

}

}

int main()

{

bintree\_class bintree;

bintree.insert\_node('A');

bintree.insert\_node('B');

bintree.insert\_node('C');

bintree.insert\_node('D');

bintree.insert\_node('E');

bintree.insert\_node('F');

bintree.insert\_node('G');

cout<<"Inorder traversal:"<<endl;

bintree.inorder\_seq();

cout<<endl<<"Postorder traversal:"<<endl;

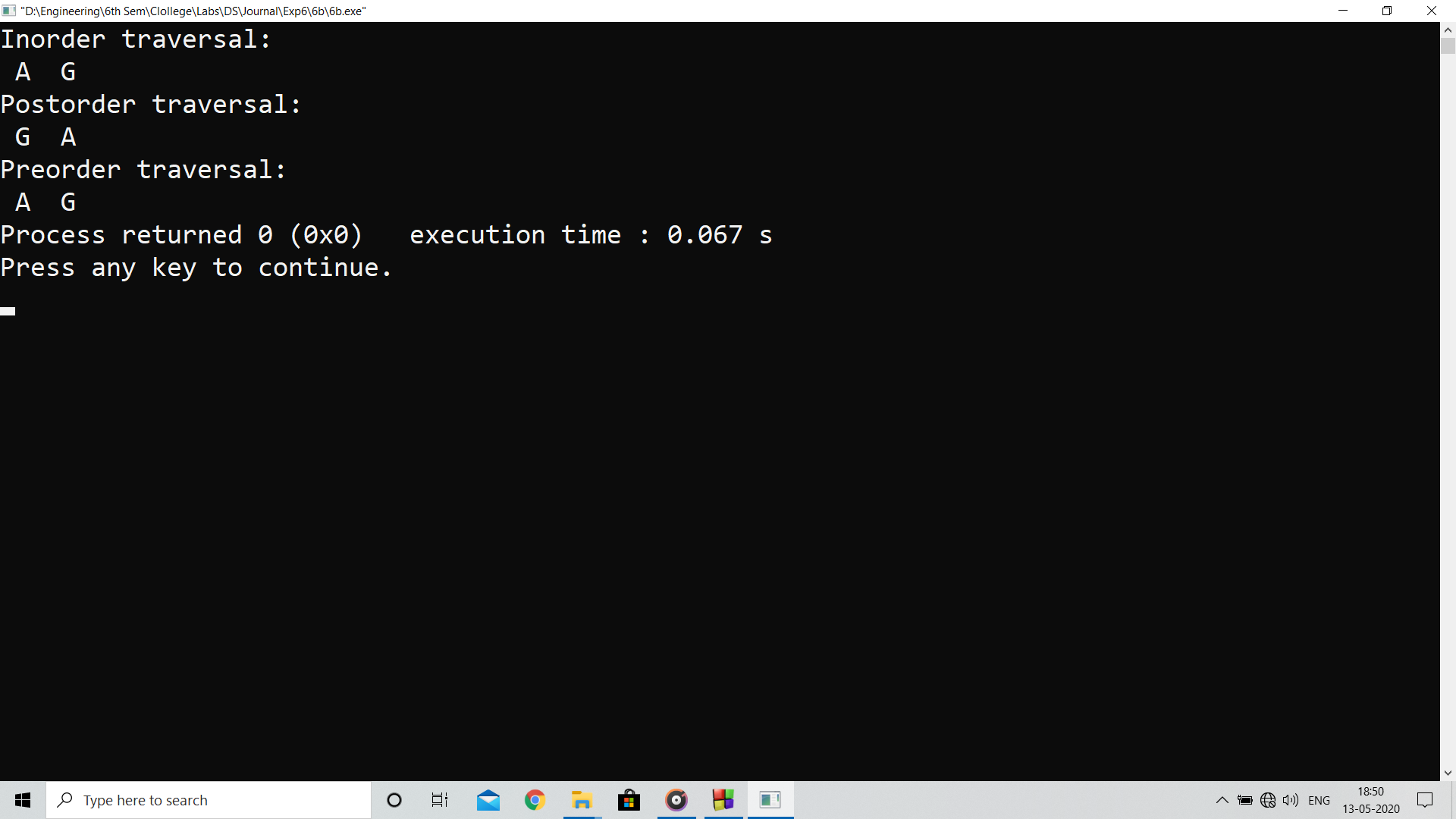
bintree.postorder\_seq();

cout<<endl<<"Preorder traversal:"<<endl;

bintree.preorder\_seq();

}

**Output:**

****

**Analysis:**

Inorder() function can give the ascending order of the values in the binary tree and preorder() and postorder() as per requirement

**Limitations:**

Inserting new element after many number of node is a costly process as insert() function travels through complete binary tree