Data Structuers and algorithms (CS09203)

Lab Report

Name:	Irtaza Kashir Raja
Registration #:	SEU-F16-127
Lab Report #:	09
Dated:	16-04-2018
Submitted To:	Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 9
DFS Graph and its representationsl

Objective
The objective of this session is to show the representation of graphs using C++.

Software Tool
1. Code Blocks with GCC compiler.

1 Theory

Depth First Traversal (or Search) for a graph is similar to Depth First Traversal of a tree. The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a boolean visited array.

2 Task

2.1 Task 1

Impeiment Depth First Traversal (or Search) for a graph.

2.2 Procedure: Task 1

```
#include<iostream>
using namespace std;
struct node{
        char data;
        struct node* left;
        struct node* right;
};
```

```
void Preorder(struct node *root){
        if(root == NULL)        return;
        cout<<root->data<<" ";
        Preorder(root->left);
        Preorder(root->right);
}

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

void Postorder(struct node *root){
        if(root == NULL)        return;
        Postorder(root->left);
        Postorder(root->right);
        cout<<root->data<<" ";
}

node* Insert(node *root, char data){
        if(root == NULL){
                root = new node();
                root->data = data;
                root->left = root->right = NULL;
        }
        else if(data <= root->data)
                root->left = Insert(root->left, data);
        else
                root->right = Insert(root->right, data);
        return root;
}

int main(){
        node* root = NULL;
        root = Insert(root, 'M');       root = Insert(root, 'B');
        root = Insert(root, 'Q');       root = Insert(root, 'Z');
        root = Insert(root, 'A');       root = Insert(root, 'C');
```

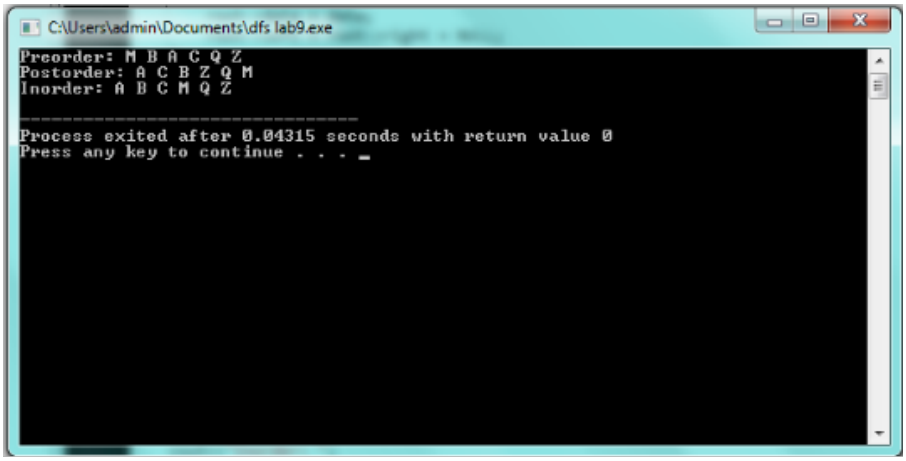

Figure 1: output

```
        cout<<"Preorder:  ";
        Preorder(root);
        cout<<"\n";
        cout<<"Postorder:  ";
        Postorder(root);
        cout<<"\n";
        cout<<"Inorder:  ";
        Inorder(root);
        cout<<"\n";
}
```