Experiment – 3.3

Student Name: Rohit Kumar Mahato	**UID: 21BCS7480**
Branch: CSE	**Section/Group: 717-A**
Semester: 3rd	**Date of Performance: 04/11/2022**
Subject Name: Data structure	**SubjectCode:21CSH-211**

1. **AIM:-** Write a program to illustrate the traversal of graph using
   a) Breadth-first search
   b) Depth-first search
2. **ALGORITHM:**

BFS

```
Step 1: SET STATUS = 1 (ready state) for each node in G

Step 2: Enqueue the starting node A and set its STATUS = 2 (waiting state)

Step 3: Repeat Steps 4 and 5 until QUEUE is empty

Step 4: Dequeue a node N. Process it and set its STATUS = 3 (processed state).

Step 5: Enqueue all the neighbours of N that are in the ready state (whose STATUS = 1) and
set their STATUS = 2(waiting state)

[END OF LOOP]

Step 6: EXIT
```

DFS

Step 1: SET STATUS = 1 (ready state) for each node in G

Step 2: Push the starting node A on the stack and set its STATUS = 2 (waiting state)

Step 3: Repeat Steps 4 and 5 until STACK is empty

Step 4: Pop the top node N. Process it and set its STATUS = 3 (processed state)

Step 5: Push on the stack all the neighbors of N that are in the ready state (whose STATUS = 1) and set their STATUS = 2 (waiting state)

[END OF LOOP]

Step 6: EXIT

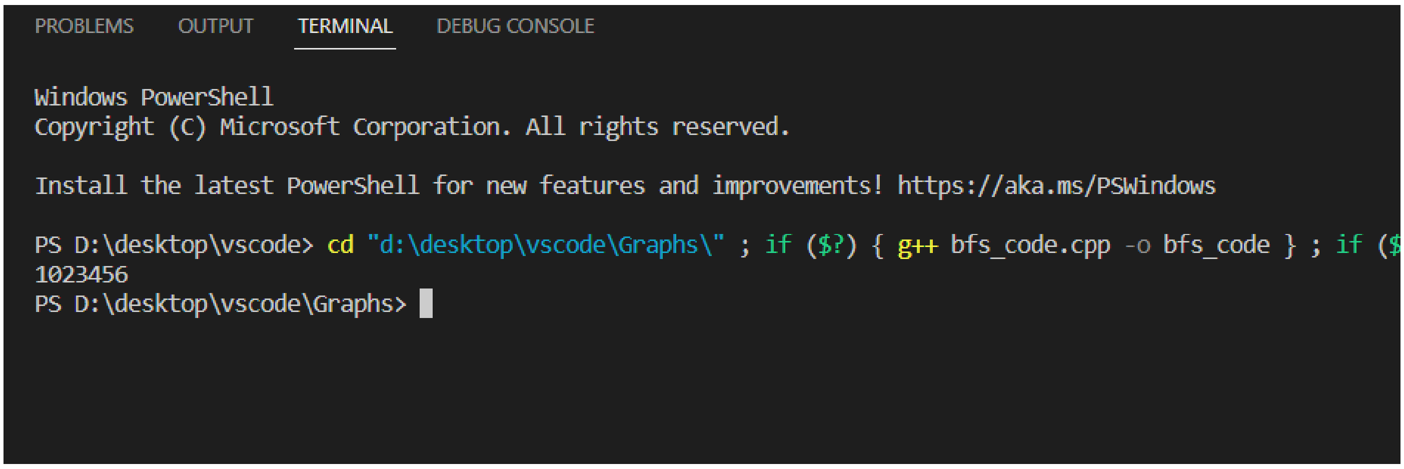
3. Program Code

BFS CODE

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

struct queue
{
    int size;
    int f;
    int r;
    int* arr;
};


int isEmpty(struct queue *q){
    if(q->r==q->f){
        return 1;
    }
    return 0;
}

int isFull(struct queue *q){
    if(q->r==q->size-1){
        return 1;
```

```cpp
    }
    return 0;
}

void enqueue(struct queue *q, int val){
    if(isFull(q)){
        cout<<"This Queue is full\n";
    }
    else{
        q->r++;
        q->arr[q->r] = val;
            }
}

int dequeue(struct queue *q){
    int a = -1;
    if(isEmpty(q)){
        cout<<"This Queue is empty\n";
    }
    else{
        q->f++;
        a = q->arr[q->f];
    }
    return a;
}

int main(){

    struct queue q;
    q.size = 400;
    q.f = q.r = 0;
    q.arr = (int*) malloc(q.size*sizeof(int));

    // BFS Implementation
    int node;
    int i = 1;
    int visited[7] = {0,0,0,0,0,0,0};
    int a [7][7] = {
        {0,1,1,1,0,0,0},
        {1,0,1,0,0,0,0},
        {1,1,0,1,1,0,0},
        {1,0,1,0,1,0,0},
        {0,0,1,1,0,1,1},
        {0,0,0,0,1,0,0},
        {0,0,0,0,1,0,0}
    };
    cout<<i;
    visited[i] = 1;
    enqueue(&q, i);
    while (!isEmpty(&q))
    {
        int node = dequeue(&q);
        for (int j = 0; j < 7; j++)
        {
            if(a[node][j] ==1 && visited[j] == 0){
                cout<< j;
                visited[j] = 1;
                enqueue(&q, j);
            }
        }
    }
    return 0;
}
```

Output

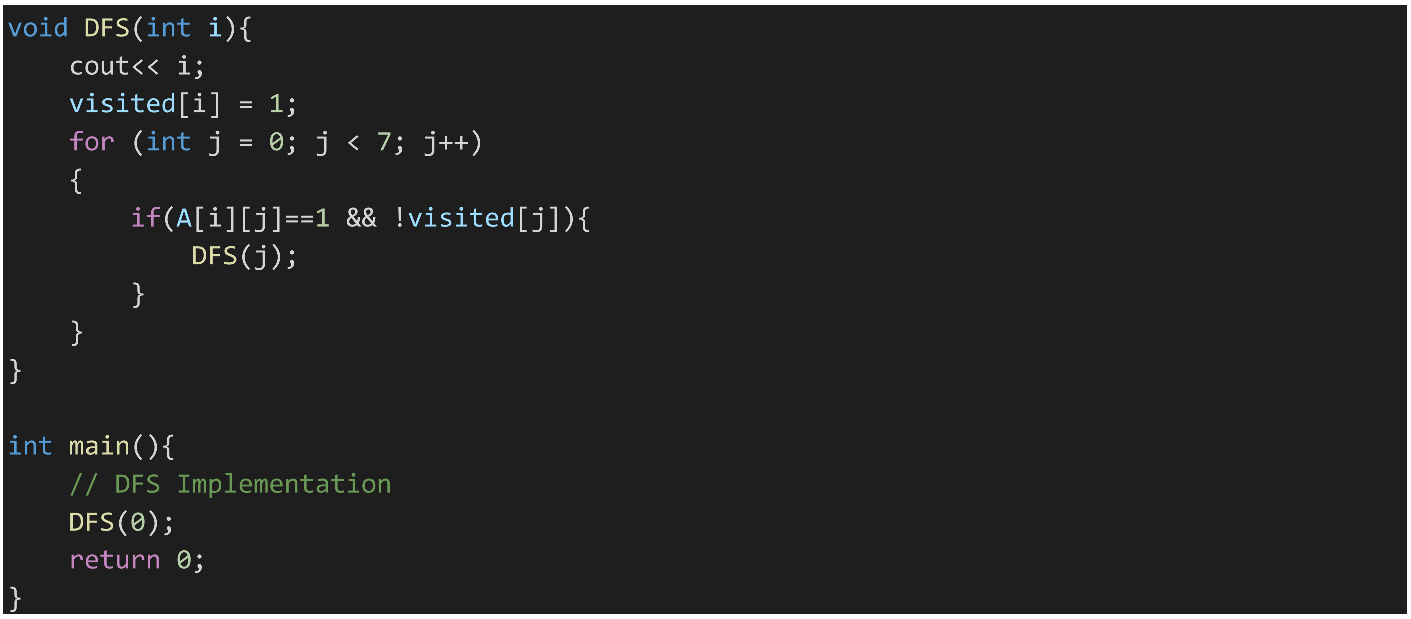


PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\desktop\vscode> cd "d:\desktop\vscode\Graphs\" ; if ($?) { g++ bfs_code.cpp -o bfs_code } ; if ($
1023456
PS D:\desktop\vscode\Graphs>
```

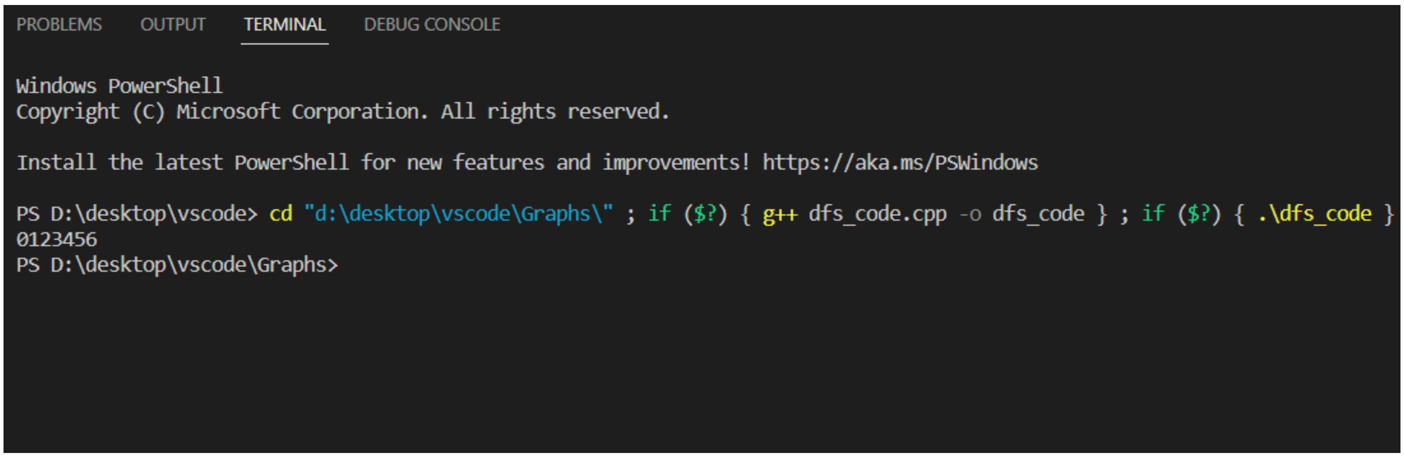
DFS CODE

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

int visited[7] = {0,0,0,0,0,0,0};
    int A [7][7] = {
        {0,1,1,1,0,0,0},
        {1,0,1,0,0,0,0},
        {1,1,0,1,1,0,0},
        {1,0,1,0,1,0,0},
        {0,0,1,1,0,1,1},
        {0,0,0,0,1,0,0},
        {0,0,0,0,1,0,0}
    };
```



```
void DFS(int i){
    cout<< i;
    visited[i] = 1;
    for (int j = 0; j < 7; j++)
    {
        if(A[i][j]==1 && !visited[j]){
            DFS(j);
        }
    }
}

int main(){
    // DFS Implementation
    DFS(0);
    return 0;
}
```

OUTPUT



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\desktop\vscode> cd "d:\desktop\vscode\Graphs\" ; if ($?) { g++ dfs_code.cpp -o dfs_code } ; if ($?) { .\dfs_code }
0123456
PS D:\desktop\vscode\Graphs>
```

Learning Outcomes (What I have learnt)

1. Learnt about Graphs and its types
2. Also learnt the various operations on Graphs