Assignment 8

Name:Soham Mahajan

Roll No:SYCOC160

Program:

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

struct AdjListNode {
    int dest;
    struct AdjListNode* next;
};

struct AdjList {
    struct AdjListNode* head;
};

class Graph {
private:
    int V;
    struct AdjList* total;
    int* visited_array;

public:
    Graph(int no_of_vertices) {
```

```
    V = no_of_vertices;
    total = new struct AdjList[V];
    visited_array = new int[V];

    for (int i = 0; i < V; i++) {
        total[i].head = nullptr;
        visited_array[i] = 0;
    }
}

void addEdge(int src, int dest) {
    struct AdjListNode* new_node = new struct AdjListNode;
    new_node->dest = dest;
    new_node->next = nullptr;

    struct AdjListNode* temp = total[src].head;
    if (temp == nullptr)
        total[src].head = new_node;
    else {
        while (temp->next != nullptr)
            temp = temp->next;
        temp->next = new_node;
    }

    struct AdjListNode* nn2 = new struct AdjListNode;
    nn2->dest = src;
```

```
    nn2->next = nullptr;
    temp = total[dest].head;
    if (temp == nullptr)
        total[dest].head = nn2;
    else {
        while (temp->next != nullptr)
            temp = temp->next;
        temp->next = nn2;
    }
}

void printAdjList() {
    cout << "printing" << endl;
    for (int i = 0; i < V; i++) {
        struct AdjListNode* temp = total[i].head;
        while (temp != nullptr) {
            cout << " ->" << temp->dest;
            temp = temp->next;
        }
        cout << endl;
    }
}

void bfs(int start) {
    queue<int> Q;
    int visited[V] = {0};
```

```
    Q.push(start);
    visited[start] = 1;
    while (!Q.empty()) {
        int printVertex = Q.front();
        Q.pop();
        cout << printVertex << endl;
        struct AdjListNode* temp = total[printVertex].head;
        while (temp != nullptr) {
            if (visited[temp->dest] == 0) {
                Q.push(temp->dest);
                visited[temp->dest] = 1;
            }
            temp = temp->next;
        }
    }
}

void dfs(int v) {
    cout << v << endl;
    visited_array[v] = 1;
    struct AdjListNode* temp = total[v].head;
    while (temp != nullptr) {
        if (visited_array[temp->dest] == 0)
            dfs(temp->dest);
        temp = temp->next;
    }
```

```
    }
};

int main() {
    cout << "Graph representation using adjacency list.." << endl;
    Graph g(6);
    cout << "Adding edge" << endl;
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(0, 3);
    g.addEdge(1, 4);
    g.addEdge(1, 5);
    g.printAdjList();
    cout << "BFS..." << endl;
    g.bfs(1);
    cout << "DFS..." << endl;
    g.dfs(3);
    cout << "Done";
    return 0;
}
```

Output:

```
/tmp/Za3hXzV5bg.o
Graph representation using adjacency list..
Adding edge
printing
 ->1 ->2 ->3
```

->0 ->4 ->5

->0

->0

->1

->1

BFS...

1

0

4

5

2

3

DFS...

3

0

1

4

5

2

Done