**DATASTRUCTURES**

**PROGRAMS:**

**1.Depth First Search**

*#include <stdio.h>*

*#include <stdlib.h>*

*struct Node {*

*int vertex;*

*struct Node\* next;*

*};*

*struct Graph {*

*int numVertices;*

*struct Node\*\* adjLists;*

*int\* visited;*

*};*

*struct Node\* createNode(int v) {*

*struct Node\* newNode = malloc(sizeof(struct Node));*

*newNode->vertex = v;*

*newNode->next = NULL;*

*return newNode;*

*}*

*struct Graph\* createGraph(int vertices) {*

*struct Graph\* graph = malloc(sizeof(struct Graph));*

*graph->numVertices = vertices;*

*graph->adjLists = malloc(vertices \* sizeof(struct Node\*));*

*graph->visited = malloc(vertices \* sizeof(int));*

*for (int i = 0; i < vertices; i++) {*

*graph->adjLists[i] = NULL;*

*graph->visited[i] = 0;*

*}*

*return graph;*

*}*

*void addEdge(struct Graph\* graph, int src, int dest) {*

*struct Node\* newNode = createNode(dest);*

*newNode->next = graph->adjLists[src];*

*graph->adjLists[src] = newNode;*

*newNode = createNode(src);*

*newNode->next = graph->adjLists[dest];*

*graph->adjLists[dest] = newNode;*

*}*

*void DFS(struct Graph\* graph, int vertex) {*

*struct Node\* adjList = graph->adjLists[vertex];*

*struct Node\* temp = adjList;*

*graph->visited[vertex] = 1;*

*printf("Visited %d\n", vertex);*

*while (temp != NULL) {*

*int connectedVertex = temp->vertex;*

*if (graph->visited[connectedVertex] == 0) {*

*DFS(graph, connectedVertex);*

*}*

*temp = temp->next;*

*}*

*}*

*int main() {*

*struct Graph\* graph = createGraph(4);*

*addEdge(graph, 0, 1);*

*addEdge(graph, 0, 2);*

*addEdge(graph, 1, 2);*

*addEdge(graph, 2, 3);*

*DFS(graph, 0);*

*return 0;*

*}*

*OUTPUT:*

*Visited 0*

*Visited 2*

*Visited 3*

*Visited 1*

**2.Breadth First Search**

*#include <stdio.h>*

*#include <stdlib.h>*

*struct Node {*

*int vertex;*

*struct Node\* next;*

*};*

*struct Graph {*

*int numVertices;*

*struct Node\*\* adjLists;*

*int\* visited;*

*};*

*struct Queue {*

*int items[100];*

*int front;*

*int rear;*

*};*

*struct Queue\* createQueue() {*

*struct Queue\* queue = malloc(sizeof(struct Queue));*

*queue->front = -1;*

*queue->rear = -1;*

*return queue;*

*}*

*int isEmpty(struct Queue\* queue) {*

*if (queue->rear == -1)*

*return 1;*

*else*

*return 0;*

*}*

*void enqueue(struct Queue\* queue, int value) {*

*if (queue->rear == 99)*

*printf("Queue is full\n");*

*else {*

*if (queue->front == -1)*

*queue->front = 0;*

*queue->rear++;*

*queue->items[queue->rear] = value;*

*}*

*}*

*int dequeue(struct Queue\* queue) {*

*int item;*

*if (isEmpty(queue)) {*

*printf("Queue is empty\n");*

*item = -1;*

*} else {*

*item = queue->items[queue->front];*

*queue->front++;*

*if (queue->front > queue->rear) {*

*queue->front = queue->rear = -1;*

*}*

*}*

*return item;*

*}*

*struct Node\* createNode(int v) {*

*struct Node\* newNode = malloc(sizeof(struct Node));*

*newNode->vertex = v;*

*newNode->next = NULL;*

*return newNode;*

*}*

*struct Graph\* createGraph(int vertices) {*

*struct Graph\* graph = malloc(sizeof(struct Graph));*

*graph->numVertices = vertices;*

*graph->adjLists = malloc(vertices \* sizeof(struct Node\*));*

*graph->visited = malloc(vertices \* sizeof(int));*

*for (int i = 0; i < vertices; i++) {*

*graph->adjLists[i] = NULL;*

*graph->visited[i] = 0;*

*}*

*return graph;*

*}*

*void addEdge(struct Graph\* graph, int src, int dest) {*

*struct Node\* newNode = createNode(dest);*

*newNode->next = graph->adjLists[src];*

*graph->adjLists[src] = newNode;*

*newNode = createNode(src);*

*newNode->next = graph->adjLists[dest];*

*graph->adjLists[dest] = newNode;*

*}*

*void BFS(struct Graph\* graph, int startVertex) {*

*struct Queue\* queue = createQueue();*

*graph->visited[startVertex] = 1;*

*enqueue(queue, startVertex);*

*while (!isEmpty(queue)) {*

*int currentVertex = dequeue(queue);*

*printf("Visited %d\n", currentVertex);*

*struct Node\* temp = graph->adjLists[currentVertex];*

*while (temp) {*

*int adjVertex = temp->vertex;*

*if (graph->visited[adjVertex] == 0) {*

*graph->visited[adjVertex] = 1;*

*enqueue(queue, adjVertex);*

*}*

*temp = temp->next;*

*}*

*}*

*}*

*int main() {*

*struct Graph\* graph = createGraph(4);*

*addEdge(graph, 0, 1);*

*addEdge(graph, 0, 2);*

*addEdge(graph, 1, 2);*

*addEdge(graph, 2, 3);*

*BFS(graph, 0);*

*return 0;*

*}*

*OUTPUT:*

*Visited 0*

*Visited 2*

*Visited 1*

*Visited 3*