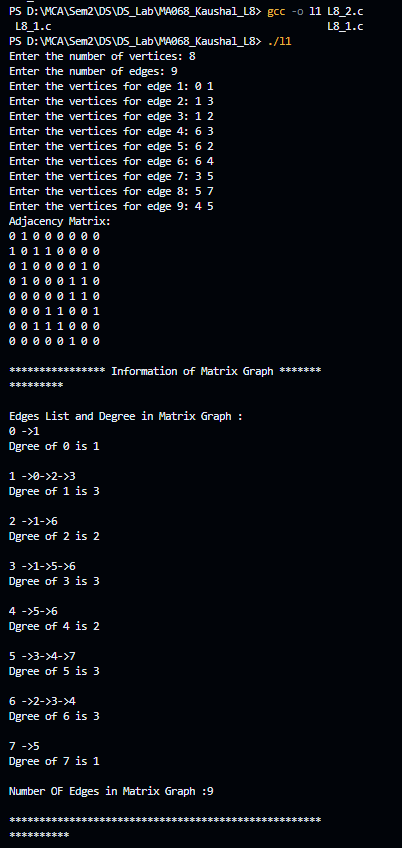
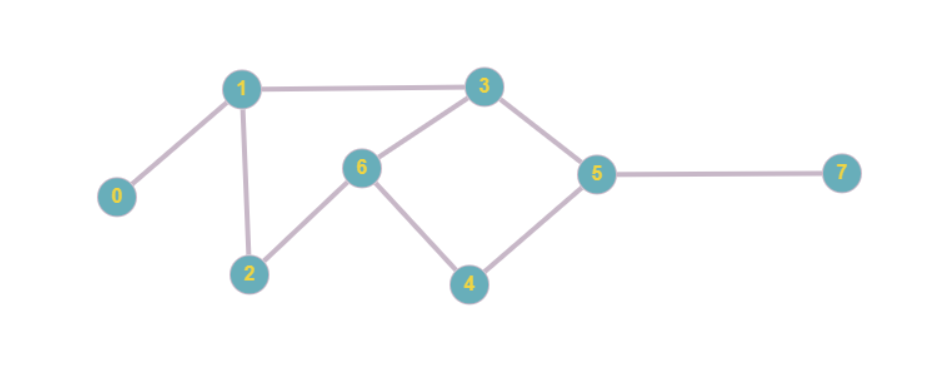
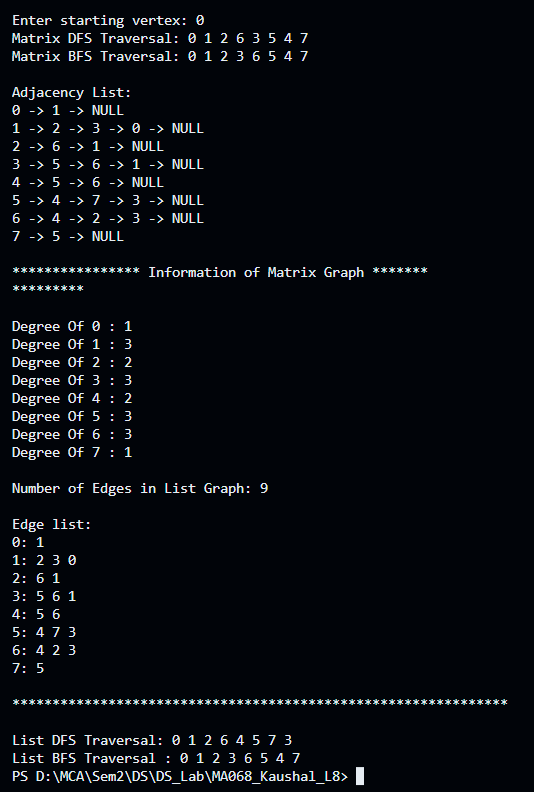
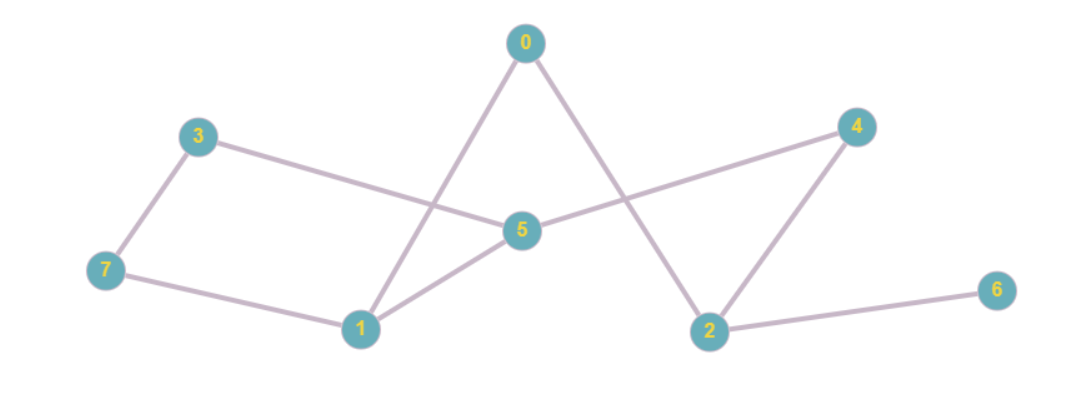
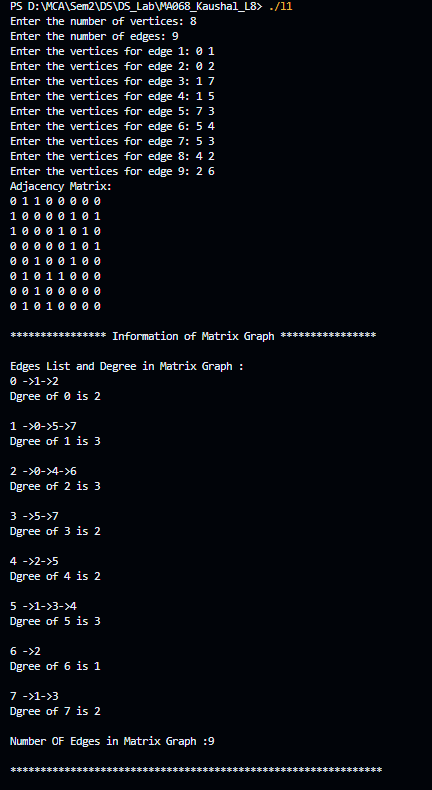
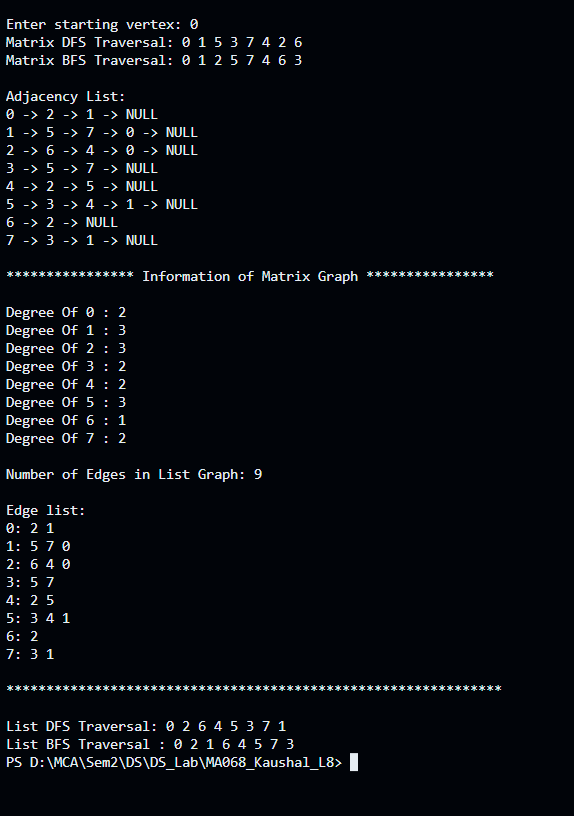
1. **Write a program to implement an undirected graph with the following.** 
   * **Create an adjacency matrix.**
   * **Create an adjacency List.**
   * **Print the information of the graph such as number of edges, edges list, degree of each vertex. (using both matrix and list)**
   * **implement traversal of graph using DFS (using both matrix and list)**
   * **implement traversal of graph using BFS. (using both matrix and list)**

**Code :**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #define MAX 100  int visited[MAX],vertex;  void initializeVisitedArray(int vertices){  for( vertex = 0; vertex < vertices; vertex++) {  visited[vertex] = 0;  // printf("\n%d",visited[vertex]);  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Matrix \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  int adjMatrix[MAX][MAX];  int vertices, edges;  void addMatrix(int u, int v){  adjMatrix[u][v] = 1;  adjMatrix[v][u] = 1;  }  void printAdjMatrix() {    printf("Adjacency Matrix:\n");  for(int i = 0; i < vertices; i++) {  for(int j = 0; j < vertices; j++) {  printf("%d ", adjMatrix[i][j]);  }  printf("\n");  }  }  void DFSAdjMatrix(int vertex, int visited[])  {  visited[vertex] = 1;  printf("%d ", vertex);  for(int i = 0; i < vertices; i++) {  if (adjMatrix[vertex][i] == 1 && visited[i] == 0) {  DFSAdjMatrix(i, visited);  }  }  }  void BFSAdjMatrix(int vertex, int visited[]) {  int queue[MAX];  int front = 0, rear = 0, i, current;  for(i = 0; i < vertices; i++) {  visited[i] = 0;  }  visited[vertex] = 1;  queue[rear++] = vertex;  while(front != rear) {    current = queue[front++];  printf("%d ", current);    for(i = 0; i < vertices; i++) {  if (adjMatrix[current][i] == 1 && visited[i] == 0) {  visited[i] = 1;  queue[rear++] = i;  }  }  }  }  void infomationOfMatrixGraph(int vertices){  int edges=0,dgree;  printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Information of Matrix Graph \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf("\nEdges List and Degree in Matrix Graph :\n");    for(int i=0; i<vertices; i++){  printf("%d ",i);  dgree =0;  for(int j=0; j<vertices; j++){  if(adjMatrix[i][j]==1)  { printf("->%d",j);  edges++; //Count edges  dgree++; //Count Dgree for Each Vertex;  }  }  printf("\nDgree of %d is %d\n",i,dgree);  printf("\n");  }  printf("Number OF Edges in Matrix Graph :%d\n",(edges/2));  printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* List \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  typedef struct node {  int vertex; //Vertext = Data element  struct node \*next;  } node;  node \*adjList[MAX];  void addEdge(int v, int e) {  node \*newNode = (node \*)malloc(sizeof(node));    newNode->vertex = e;  newNode->next = adjList[v];  adjList[v] = newNode;  }  void printAdjList() {  int i;  node \*temp;  printf("\nAdjacency List:\n");  for (i = 0; i < vertices; i++) {  printf("%d -> ", i);  temp = adjList[i];  while (temp != NULL) {  printf("%d -> ", temp->vertex);  temp = temp->next;  }  printf("NULL\n");  }  }  void DFSAdjList(int vertex, int visited[]) {  visited[vertex] = 1;  printf("%d ", vertex);  node \*temp = adjList[vertex];  while (temp != NULL) {  if (!visited[temp->vertex]) {  DFSAdjList(temp->vertex, visited);  }  temp = temp->next;  }  }  void BFSAdjList(int start, int visited[]) {  int queue[MAX], front = 0, rear = 0;  visited[start] = 1;  queue[rear++] = start;  printf("%d ", start);  while (front < rear) {  int vertex = queue[front++];  node \*temp = adjList[vertex];  while (temp != NULL) {  if (!visited[temp->vertex]) {  visited[temp->vertex] = 1;  printf("%d ", temp->vertex);  queue[rear++] = temp->vertex;  }  temp = temp->next;  }  }  }  void informationOfListGraph(int vertices) {  int numEdges = 0;  printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Information of Matrix Graph \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");      for(int i = 0; i < vertices; i++) {  node \*current = adjList[i];  int degree=0;    while(current != NULL) {  numEdges++;  degree++;  current = current->next;  }  printf("\nDegree Of %d : %d ",i,degree);  }  printf("\n\nNumber of Edges in List Graph: %d\n", numEdges / 2);      printf("\nEdge list:\n");  for(int i = 0; i < vertices; i++) {  printf("%d: ", i);  node \*current = adjList[i];  while(current != NULL) {  printf("%d ", current->vertex);  current = current->next;  }  printf("\n");  }    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    }  int main() {      // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Same Input \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  int v,e;  printf("Enter the number of vertices: ");  scanf("%d", &vertices);  printf("Enter the number of edges: ");  scanf("%d", &edges);    for (int i = 0; i < vertices; i++) {  adjList[i] = NULL;  }  for (int j = 0; j < edges; j++) {  printf("Enter the vertices for edge %d: ", j+1);  scanf("%d %d", &v, &e);  addEdge(v, e); //For List  addEdge(e, v);  addMatrix(v, e); //For Matrix  }  printAdjMatrix();  infomationOfMatrixGraph(vertices);    printf("\nEnter starting vertex: ");  scanf("%d", &vertex);  //DFS  for(int i = 0; i < vertices; i++) {  visited[i] = 0;  // printf("\n%d",visited[i]);  }  //initializeVisitedArray(vertices);  printf("Matrix DFS Traversal: ");  DFSAdjMatrix(vertex, visited);  printf("\n");  //BFS  for(int i = 0; i < vertices; i++) {  visited[i] = 0;  // printf("\n%d",visited[i]);  }  //initializeVisitedArray(vertices);  printf("Matrix BFS Traversal: ");  BFSAdjMatrix(vertex, visited);  printf("\n");    printAdjList();  informationOfListGraph(vertices);    // DFS  initializeVisitedArray(vertices);  printf("\nList DFS Traversal: ");  for (vertex = 0; vertex < vertices; vertex++) {  if (!visited[vertex]) {  DFSAdjList(vertex, visited);  }  }  // BFS  initializeVisitedArray(vertices);  printf("\nList BFS Traversal : ");  for (vertex = 0; vertex < vertices; vertex++) {  if (!visited[vertex]) {  BFSAdjList(vertex, visited);  }  }    return 0;  } |

**Output-1: **

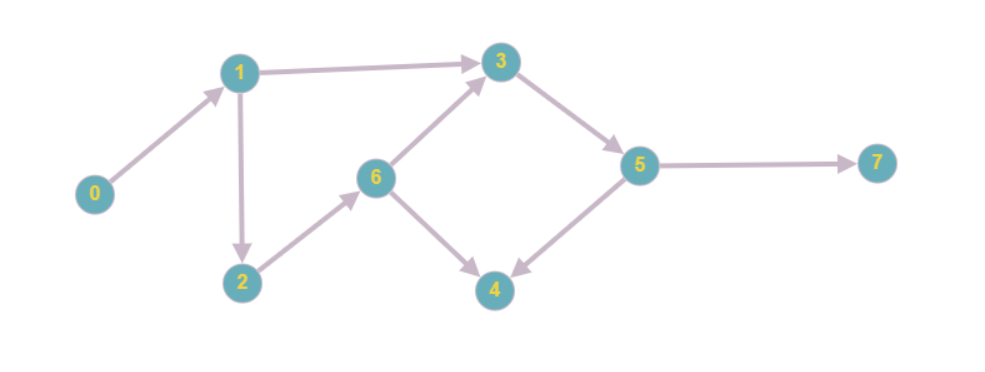
**Output-2:**

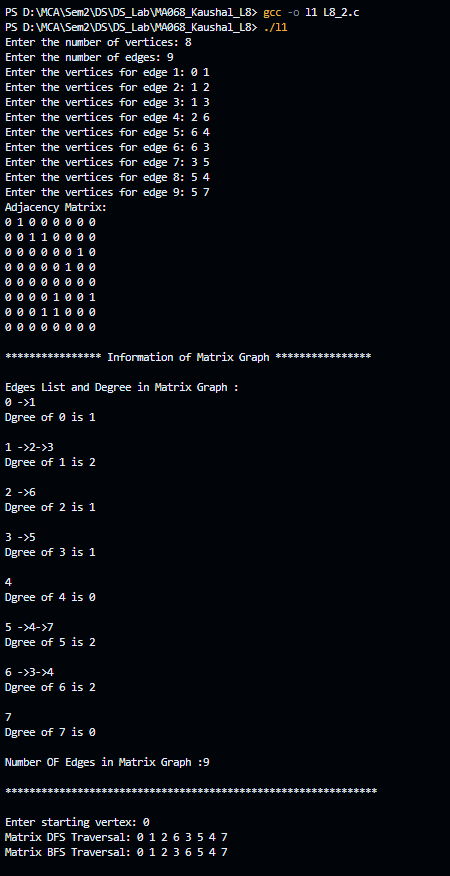
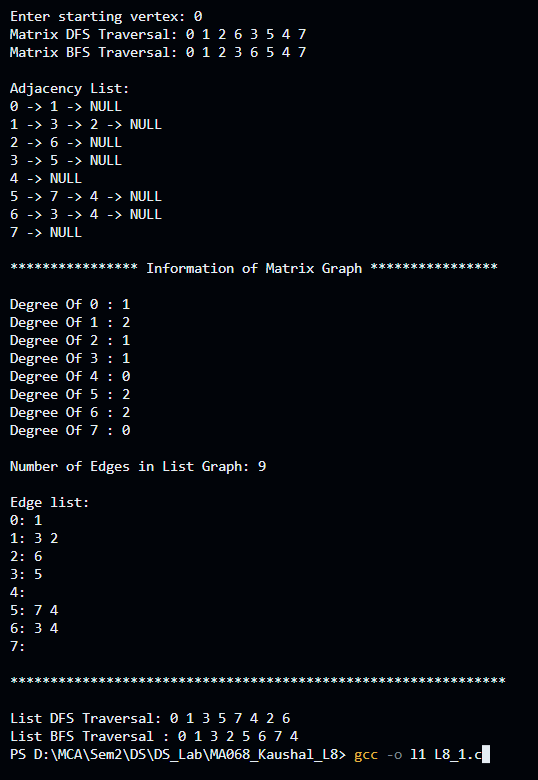
** **

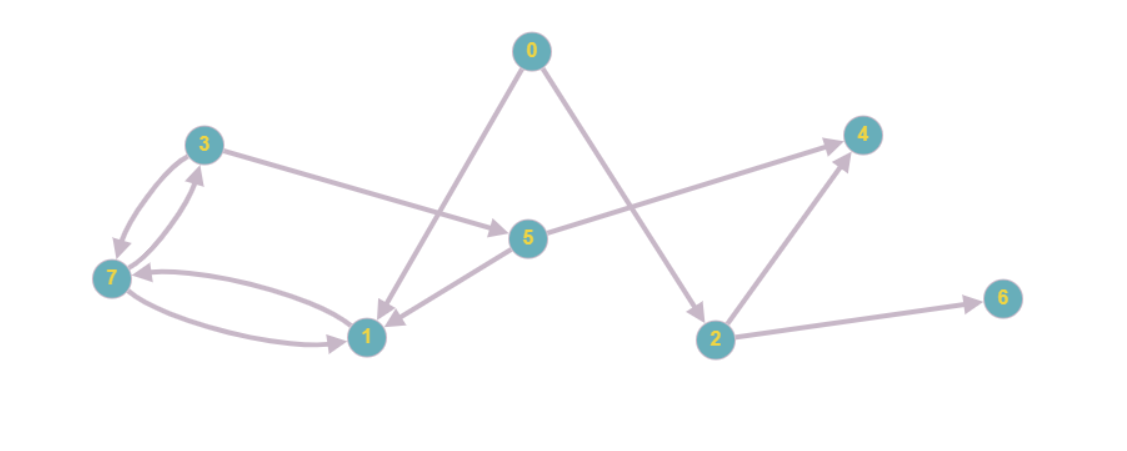
1. **Write a program to implement an directed graph with the following.** 
   * **Create an adjacency matrix.**
   * **Create an adjacency List.**
   * **Print the information of the graph such as number of edges, edges list, degree of each vertex. (using both matrix and list)**
   * **implement traversal of graph using DFS (using both matrix and list)**
   * **implement traversal of graph using BFS. (using both matrix and list)**

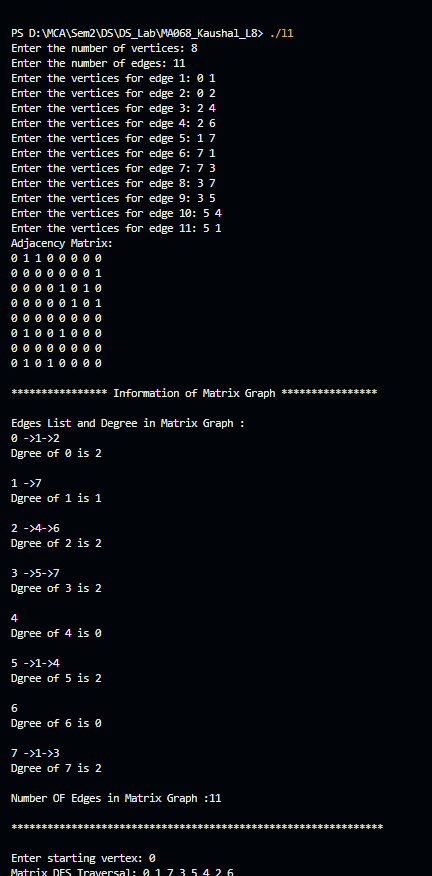
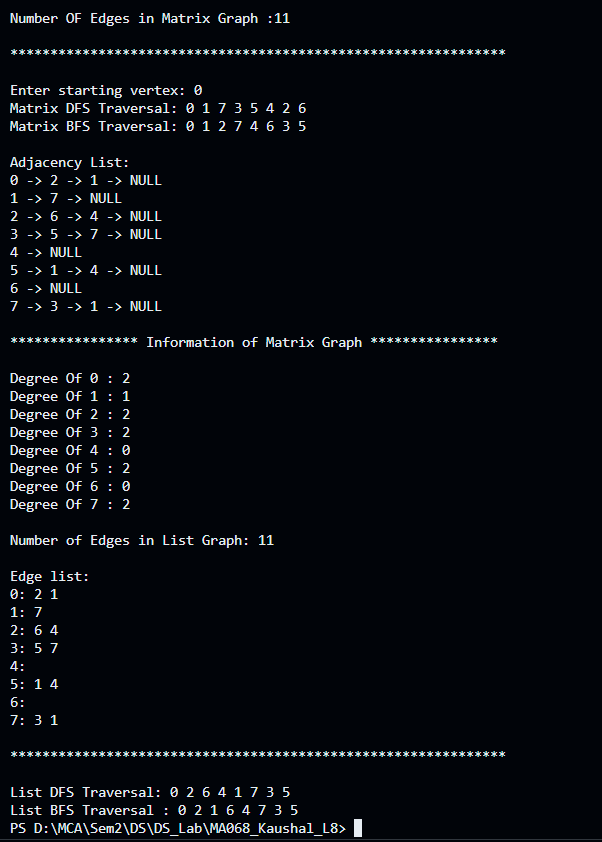
**Code:**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #define MAX 100  int visited[MAX],vertex;  void initializeVisitedArray(int vertices){  for( vertex = 0; vertex < vertices; vertex++) {  visited[vertex] = 0;  // printf("\n%d",visited[vertex]);  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Matrix \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  int adjMatrix[MAX][MAX];  int vertices, edges;  void addMatrix(int u, int v){  adjMatrix[u][v] = 1;    }  void printAdjMatrix() {    printf("Adjacency Matrix:\n");  for(int i = 0; i < vertices; i++) {  for(int j = 0; j < vertices; j++) {  printf("%d ", adjMatrix[i][j]);  }  printf("\n");  }  }  void DFSAdjMatrix(int vertex, int visited[]) {    visited[vertex] = 1;  printf("%d ", vertex);    for(int i = 0; i < vertices; i++) {  if (adjMatrix[vertex][i] == 1 && visited[i] == 0) {  DFSAdjMatrix(i, visited);  }  }  }  void BFSAdjMatrix(int vertex, int visited[]) {  int queue[MAX];  int front = 0, rear = 0, i, current;  for(i = 0; i < vertices; i++) {  visited[i] = 0;  }  visited[vertex] = 1;  queue[rear++] = vertex;  while(front != rear) {    current = queue[front++];  printf("%d ", current);    for(i = 0; i < vertices; i++) {  if (adjMatrix[current][i] == 1 && visited[i] == 0) {  visited[i] = 1;  queue[rear++] = i;  }  }  }  }  void infomationOfMatrixGraph(int vertices){  int edges=0,dgree;  printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Information of Matrix Graph \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  printf("\nEdges List and Degree in Matrix Graph :\n");    for(int i=0; i<vertices; i++){  printf("%d ",i);  dgree =0;  for(int j=0; j<vertices; j++){  if(adjMatrix[i][j]==1)  { printf("->%d",j);  edges++; //Count edges  dgree++; //Count Dgree for Each Vertex;  }  }  printf("\nDgree of %d is %d\n",i,dgree);  printf("\n");  }  printf("Number OF Edges in Matrix Graph :%d\n",(edges));  printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* List \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  typedef struct node {  int vertex; //Vertext = Data element  struct node \*next;  } node;  node \*adjList[MAX];  void addEdge(int v, int e) {  node \*newNode = (node \*)malloc(sizeof(node));    newNode->vertex = e;  newNode->next = adjList[v];  adjList[v] = newNode;  }  void printAdjList() {  int i;  node \*temp;  printf("\nAdjacency List:\n");  for (i = 0; i < vertices; i++) {  printf("%d -> ", i);  temp = adjList[i];  while (temp != NULL) {  printf("%d -> ", temp->vertex);  temp = temp->next;  }  printf("NULL\n");  }  }  void DFSAdjList(int vertex, int visited[]) {  visited[vertex] = 1;  printf("%d ", vertex);  node \*temp = adjList[vertex];  while (temp != NULL) {  if (!visited[temp->vertex]) {  DFSAdjList(temp->vertex, visited);  }  temp = temp->next;  }  }  void BFSAdjList(int start, int visited[]) {  int queue[MAX], front = 0, rear = 0;  visited[start] = 1;  queue[rear++] = start;  printf("%d ", start);  while (front < rear) {  int vertex = queue[front++];  node \*temp = adjList[vertex];  while (temp != NULL) {  if (!visited[temp->vertex]) {  visited[temp->vertex] = 1;  printf("%d ", temp->vertex);  queue[rear++] = temp->vertex;  }  temp = temp->next;  }  }  }  void informationOfListGraph(int vertices) {  int numEdges = 0;  printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Information of Matrix Graph \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");      for(int i = 0; i < vertices; i++) {  node \*current = adjList[i];  int degree=0;    while(current != NULL) {  numEdges++;  degree++;  current = current->next;  }  printf("\nDegree Of %d : %d ",i,degree);  }  printf("\n\nNumber of Edges in List Graph: %d\n", numEdges );      printf("\nEdge list:\n");  for(int i = 0; i < vertices; i++) {  printf("%d: ", i);  node \*current = adjList[i];    while(current != NULL) {  printf("%d ", current->vertex);  current = current->next;  }  printf("\n");  }    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");    }  int main() {      // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Same Input \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  int v,e;  printf("Enter the number of vertices: ");  scanf("%d", &vertices);  printf("Enter the number of edges: ");  scanf("%d", &edges);    for (int i = 0; i < vertices; i++) {  adjList[i] = NULL;  }  for (int j = 0; j < edges; j++) {  printf("Enter the vertices for edge %d: ", j+1);  scanf("%d %d", &v, &e);  addEdge(v, e); //For List    addMatrix(v, e); //For Matrix  }  printAdjMatrix();  infomationOfMatrixGraph(vertices);    printf("\nEnter starting vertex: ");  scanf("%d", &vertex);  //DFS  for(int i = 0; i < vertices; i++) {  visited[i] = 0;  // printf("\n%d",visited[i]);  }  //initializeVisitedArray(vertices);  printf("Matrix DFS Traversal: ");  DFSAdjMatrix(vertex, visited);  printf("\n");  //BFS  for(int i = 0; i < vertices; i++) {  visited[i] = 0;  // printf("\n%d",visited[i]);  }  //initializeVisitedArray(vertices);  printf("Matrix BFS Traversal: ");  BFSAdjMatrix(vertex, visited);  printf("\n");    printAdjList();  informationOfListGraph(vertices);    // DFS  initializeVisitedArray(vertices);  printf("\nList DFS Traversal: ");    for (vertex = 0; vertex < vertices; vertex++) {  if (!visited[vertex]) {  DFSAdjList(vertex, visited);  }  }  // BFS  initializeVisitedArray(vertices);  printf("\nList BFS Traversal : ");    for (vertex = 0; vertex < vertices; vertex++) {  if (!visited[vertex]) {  BFSAdjList(vertex, visited);  }  }    return 0;  } |

**Output-1:**

** **

**Output-2:**

** **