Practical – 8 Implementation of Graph

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;
      }
   }
}
```

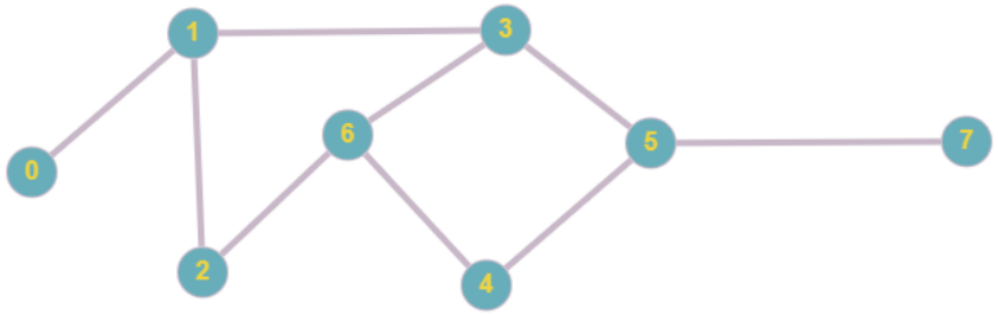
```
void informationOfListGraph(int vertices) {
   int numEdges = 0;

   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;
}
```

Practical – 8 Implementation of Graph

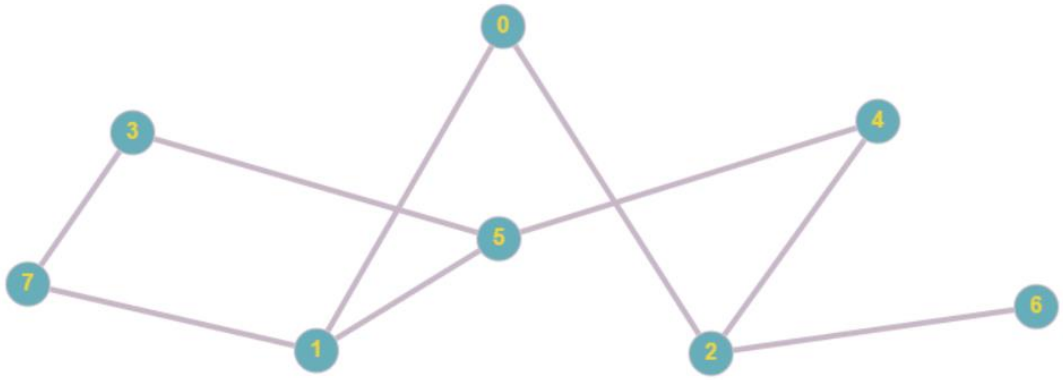
Output-1:




```
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L8> gcc -o 11 L8_2.c
 L8_1.c                                              L8_1.c
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L8> ./11
Enter the number of vertices: 8
Enter the number of edges: 9
Enter the vertices for edge 1: 0 1
Enter the vertices for edge 2: 1 3
Enter the vertices for edge 3: 1 2
Enter the vertices for edge 4: 6 3
Enter the vertices for edge 5: 6 2
Enter the vertices for edge 6: 6 4
Enter the vertices for edge 7: 3 5
Enter the vertices for edge 8: 5 7
Enter the vertices for edge 9: 4 5
Adjacency Matrix:
0 1 0 0 0 0 0 0
1 0 1 1 0 0 0 0
0 1 0 0 0 0 1 0
0 1 0 0 0 1 1 0
0 0 0 0 0 1 1 0
0 0 0 1 1 0 0 1
0 0 1 1 1 0 0 0
0 0 0 0 0 1 0 0

*************** Information of Matrix Graph *******
*********

Edges List and Degree in Matrix Graph :
0 ->1
Dgree of 0 is 1

1 ->0->2->3
Dgree of 1 is 3

2 ->1->6
Dgree of 2 is 2

3 ->1->5->6
Dgree of 3 is 3

4 ->5->6
Dgree of 4 is 2

5 ->3->4->7
Dgree of 5 is 3

6 ->2->3->4
Dgree of 6 is 3

7 ->5
Dgree of 7 is 1

Number OF Edges in Matrix Graph :9

**************************************************
**********
```

```
Enter starting vertex: 0
Matrix DFS Traversal: 0 1 2 6 3 5 4 7
Matrix BFS Traversal: 0 1 2 3 6 5 4 7

Adjacency List:
0 -> 1 -> NULL
1 -> 2 -> 3 -> 0 -> NULL
2 -> 6 -> 1 -> NULL
3 -> 5 -> 6 -> 1 -> NULL
4 -> 5 -> 6 -> NULL
5 -> 4 -> 7 -> 3 -> NULL
6 -> 4 -> 2 -> 3 -> NULL
7 -> 5 -> NULL

*************** Information of Matrix Graph *******
*********

Degree Of 0 : 1
Degree Of 1 : 3
Degree Of 2 : 2
Degree Of 3 : 3
Degree Of 4 : 2
Degree Of 5 : 3
Degree Of 6 : 3
Degree Of 7 : 1

Number of Edges in List Graph: 9

Edge list:
0: 1
1: 2 3 0
2: 6 1
3: 5 6 1
4: 5 6
5: 4 7 3
6: 4 2 3
7: 5

**************************************************************

List DFS Traversal: 0 1 2 6 4 5 7 3
List BFS Traversal : 0 1 2 3 6 5 4 7
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L8>
```

Output-2:




```
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L8> ./11
Enter the number of vertices: 8
Enter the number of edges: 9
Enter the vertices for edge 1: 0 1
Enter the vertices for edge 2: 0 2
Enter the vertices for edge 3: 1 7
Enter the vertices for edge 4: 1 5
Enter the vertices for edge 5: 7 3
Enter the vertices for edge 6: 5 4
Enter the vertices for edge 7: 5 3
Enter the vertices for edge 8: 4 2
Enter the vertices for edge 9: 2 6
Adjacency Matrix:
0 1 1 0 0 0 0 0
1 0 0 0 0 1 0 1
1 0 0 0 1 0 1 0
0 0 0 0 0 1 0 1
0 0 1 0 0 1 0 0
0 1 0 1 1 0 0 0
0 0 1 0 0 0 0 0
0 1 0 1 0 0 0 0

*************** Information of Matrix Graph ***************

Edges List and Degree in Matrix Graph :
0 ->1->2
Dgree of 0 is 2

1 ->0->5->7
Dgree of 1 is 3

2 ->0->4->6
Dgree of 2 is 3

3 ->5->7
Dgree of 3 is 2

4 ->2->5
Dgree of 4 is 2

5 ->1->3->4
Dgree of 5 is 3

6 ->2
Dgree of 6 is 1

7 ->1->3
Dgree of 7 is 2

Number OF Edges in Matrix Graph :9

************************************************************
```

```
Enter starting vertex: 0
Matrix DFS Traversal: 0 1 5 3 7 4 2 6
Matrix BFS Traversal: 0 1 2 5 7 4 6 3

Adjacency List:
0 -> 2 -> 1 -> NULL
1 -> 5 -> 7 -> 0 -> NULL
2 -> 6 -> 4 -> 0 -> NULL
3 -> 5 -> 7 -> NULL
4 -> 2 -> 5 -> NULL
5 -> 3 -> 4 -> 1 -> NULL
6 -> 2 -> NULL
7 -> 3 -> 1 -> NULL

*************** Information of Matrix Graph ***************

Degree Of 0 : 2
Degree Of 1 : 3
Degree Of 2 : 3
Degree Of 3 : 2
Degree Of 4 : 2
Degree Of 5 : 3
Degree Of 6 : 1
Degree Of 7 : 2

Number of Edges in List Graph: 9

Edge list:
0: 2 1
1: 5 7 0
2: 6 4 0
3: 5 7
4: 2 5
5: 3 4 1
6: 2
7: 3 1

************************************************************

List DFS Traversal: 0 2 6 4 5 3 7 1
List BFS Traversal : 0 2 1 6 4 5 7 3
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L8>
```

2. 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;
      }
   }
}
```

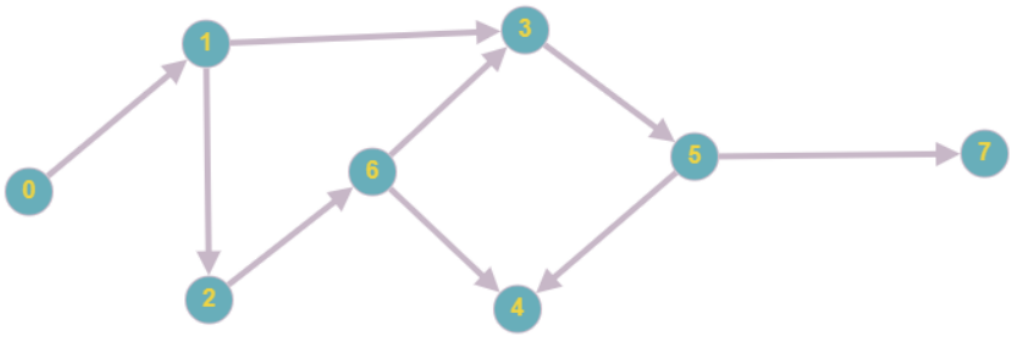
```
void informationOfListGraph(int vertices) {
   int numEdges = 0;

   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;
}
```

Practical – 8 Implementation of Graph

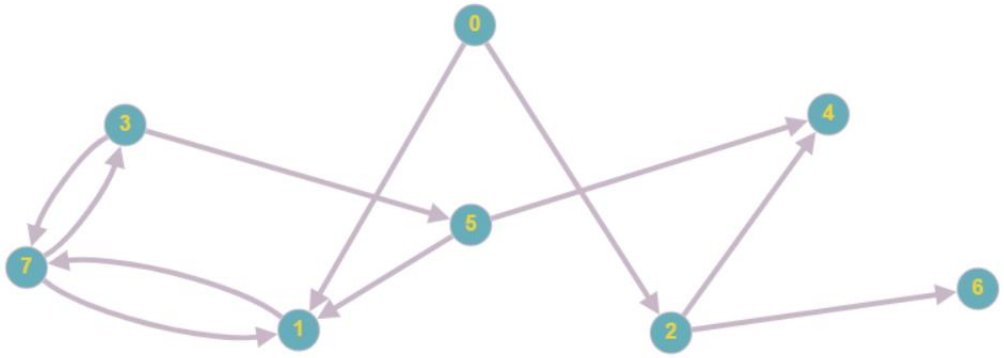
Output-1:




```
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L8> gcc -o l1 L8_2.c
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L8> ./l1
Enter the number of vertices: 8
Enter the number of edges: 9
Enter the vertices for edge 1: 0 1
Enter the vertices for edge 2: 1 2
Enter the vertices for edge 3: 1 3
Enter the vertices for edge 4: 2 6
Enter the vertices for edge 5: 6 4
Enter the vertices for edge 6: 6 3
Enter the vertices for edge 7: 3 5
Enter the vertices for edge 8: 5 4
Enter the vertices for edge 9: 5 7
Adjacency Matrix:
0 1 0 0 0 0 0 0
0 0 1 1 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 1
0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0

*************** Information of Matrix Graph ***************

Edges List and Degree in Matrix Graph :
0 ->1
Dgree of 0 is 1

1 ->2->3
Dgree of 1 is 2

2 ->6
Dgree of 2 is 1

3 ->5
Dgree of 3 is 1

4
Dgree of 4 is 0

5 ->4->7
Dgree of 5 is 2

6 ->3->4
Dgree of 6 is 2

7
Dgree of 7 is 0

Number OF Edges in Matrix Graph :9

************************************************************

Enter starting vertex: 0
Matrix DFS Traversal: 0 1 2 6 3 5 4 7
Matrix BFS Traversal: 0 1 2 3 6 5 4 7
```

```
Enter starting vertex: 0
Matrix DFS Traversal: 0 1 2 6 3 5 4 7
Matrix BFS Traversal: 0 1 2 3 6 5 4 7

Adjacency List:
0 -> 1 -> NULL
1 -> 3 -> 2 -> NULL
2 -> 6 -> NULL
3 -> 5 -> NULL
4 -> NULL
5 -> 7 -> 4 -> NULL
6 -> 3 -> 4 -> NULL
7 -> NULL

*************** Information of Matrix Graph ***************

Degree Of 0 : 1
Degree Of 1 : 2
Degree Of 2 : 1
Degree Of 3 : 1
Degree Of 4 : 0
Degree Of 5 : 2
Degree Of 6 : 2
Degree Of 7 : 0

Number of Edges in List Graph: 9

Edge list:
0: 1
1: 3 2
2: 6
3: 5
4:
5: 7 4
6: 3 4
7:

************************************************************

List DFS Traversal: 0 1 3 5 7 4 2 6
List BFS Traversal : 0 1 3 2 5 6 7 4
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L8> gcc -o l1 L8_1.c
```

Output-2:




```
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L8> ./11
Enter the number of vertices: 8
Enter the number of edges: 11
Enter the vertices for edge 1: 0 1
Enter the vertices for edge 2: 0 2
Enter the vertices for edge 3: 2 4
Enter the vertices for edge 4: 2 6
Enter the vertices for edge 5: 1 7
Enter the vertices for edge 6: 7 1
Enter the vertices for edge 7: 7 3
Enter the vertices for edge 8: 3 7
Enter the vertices for edge 9: 3 5
Enter the vertices for edge 10: 5 4
Enter the vertices for edge 11: 5 1
Adjacency Matrix:
0 1 1 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 1 0 1 0
0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 0
0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0

*************** Information of Matrix Graph ***************

Edges List and Degree in Matrix Graph :
0 ->1->2
Dgree of 0 is 2

1 ->7
Dgree of 1 is 1

2 ->4->6
Dgree of 2 is 2

3 ->5->7
Dgree of 3 is 2

4
Dgree of 4 is 0

5 ->1->4
Dgree of 5 is 2

6
Dgree of 6 is 0

7 ->1->3
Dgree of 7 is 2

Number OF Edges in Matrix Graph :11

************************************************************

Enter starting vertex: 0
Matrix DFS Traversal: 0 1 7 3 5 4 2 6
```

```
Number OF Edges in Matrix Graph :11

************************************************************

Enter starting vertex: 0
Matrix DFS Traversal: 0 1 7 3 5 4 2 6
Matrix BFS Traversal: 0 1 2 7 4 6 3 5

Adjacency List:
0 -> 2 -> 1 -> NULL
1 -> 7 -> NULL
2 -> 6 -> 4 -> NULL
3 -> 5 -> 7 -> NULL
4 -> NULL
5 -> 1 -> 4 -> NULL
6 -> NULL
7 -> 3 -> 1 -> NULL

*************** Information of Matrix Graph ***************

Degree Of 0 : 2
Degree Of 1 : 1
Degree Of 2 : 2
Degree Of 3 : 2
Degree Of 4 : 0
Degree Of 5 : 2
Degree Of 6 : 0
Degree Of 7 : 2

Number of Edges in List Graph: 11

Edge list:
0: 2 1
1: 7
2: 6 4
3: 5 7
4:
5: 1 4
6:
7: 3 1

************************************************************

List DFS Traversal: 0 2 6 4 1 7 3 5
List BFS Traversal : 0 2 1 6 4 7 3 5
PS D:\MCA\Sem2\DS\DS_Lab\MA068_Kaushal_L8>
```