Program 10:

9a) Write a program to traverse a graph using BFS method.

Observation:

```
fa) # finelude (stdfo.h)
#inelude (stdfb.h)
   # Priclude < stobool.h>
   # define MAXION 1003
   stant Quen E
       Put Ptems [Max];
        int tront;
        int rear;
   struct Queue* createQueue() f
   staut Quever 2 = (staut Queut) malloc (82206 (strutg
      9-> front = -1;
9-> rear = -1;
       veturn 9;
   boof es Empty (struct Queue * 2) {
        seturn q -) front == -1;
  void enqueue (stemt Queue* q, int value) ;
        et (a) rear == MAX-1) {
         parts ("Queunts full 'n");
       return:
}
(2 + font == -1) {
           a -> front = 0;
       9->reartt;
      q-) iterms [q->rear] = value;
```

```
At dequeue (struct Quine '9) {

Pt (esempty (2)) {

position of Gueue is empty (n'),

return -1;

}

int stem = q > Ptens (2) front;

q > front to;

if (q > front > q > rear) {

q > front = q > rear = -2;

}

roturn etem;

}

void BFS (Int graph IMPS BADIS, Ent startVerter, Ent

numVertera) {

struct Queue 'q = create Queue();

ent visited [MAN] = jog;

prints ('SFS Traversal:');

visited [start Virte] = 1;

enqueue (q, startVertex);

while (! Ps Empty (a)) {

ent current Vertex = dequeue(q);

pernts ("Ind"; current Virtex);

for (sat 8=0; senum Vertex);

if (graph fearment Vertex);

visited [i] = 1;

eq enqueue (q, e);

}

prints ('In');

free (q);
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 100
struct Queue {
    int items[MAX];
    int front;
    int rear;
};
struct Queue* createQueue() {
    struct Queue* q = (struct Queue*)malloc(sizeof(struct Queue));
    q \rightarrow front = -1;
    q \rightarrow rear = -1;
    return q;
bool isEmpty(struct Queue* q) {
    return q->front == -1;
void enqueue(struct Queue* q, int value) {
    if (q->rear == MAX - 1) {
        printf("Queue is full\n");
        return;
    if (q->front == -1) {
        q \rightarrow front = 0;
    q->rear++;
    q->items[q->rear] = value;
int dequeue(struct Queue* q) {
    if (isEmpty(q)) {
        printf("Queue is empty\n");
        return -1;
    int item = q->items[q->front];
    q->front++;
    if (q->front > q->rear) {
        q->front = q->rear = -1;
    return item;
void BFS(int graph[MAX][MAX], int startVertex, int numVertices) {
    struct Queue* q = createQueue();
```

```
int visited[MAX] = {0};
    printf("BFS Traversal: ");
    visited[startVertex] = 1;
    enqueue(q, startVertex);
    while (!isEmpty(q)) {
        int currentVertex = dequeue(q);
        printf("%d ", currentVertex);
        for (int i = 0; i < numVertices; i++) {</pre>
            if (graph[currentVertex][i] == 1 && !visited[i]) {
                visited[i] = 1;
                enqueue(q, i);
    printf("\n");
    free(q);
int main() {
    int graph[MAX][MAX], numVertices, startVertex;
    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);
    printf("Enter the adjacency matrix:\n");
    for (int i = 0; i < numVertices; i++) {</pre>
        for (int j = 0; j < numVertices; j++) {</pre>
            scanf("%d", &graph[i][j]);
        }
    printf("Enter the starting vertex: ");
    scanf("%d", &startVertex);
    BFS(graph, startVertex, numVertices);
    return 0;
```

```
Output:
```

```
Enter the number of vertices: 6
Enter the adjacency matrix:
0 1 1 0 1 1
1 0 0 1 0 1
0 1 1 1 0 0
1 1 1 0 1
1 1 1 0 0
Enter the starting vertex: 1
BFS Traversal: 1 0 3 5 2 4
```

9b) Write a program to traverse a graph using DFS method.

Observation:

```
Enter the no. of vertices: 4

Enter the adjacency matrex:

0 1 1 1

1 1 0

1 0 1 0

0 0 0 1

Enter the starting vertex:0

DFS Traversal: 0 1 2 3
```

Code:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
void DFS(int graph[MAX][MAX], int vertex, int visited[], int numVertices) {
    visited[vertex] = 1;
    printf("%d ", vertex);
    for (int i = 0; i < numVertices; i++) {</pre>
        if (graph[vertex][i] == 1 && !visited[i]) {
            DFS(graph, i, visited, numVertices);
int main() {
    int graph[MAX][MAX], numVertices, startVertex;
    int visited[MAX] = {0};
    printf("Enter the number of vertices: ");
    scanf("%d", &numVertices);
    printf("Enter the adjacency matrix:\n");
    for (int i = 0; i < numVertices; i++) {</pre>
        for (int j = 0; j < numVertices; j++) {</pre>
            scanf("%d", &graph[i][j]);
    printf("Enter the starting vertex: ");
    scanf("%d", &startVertex);
    printf("DFS Traversal: ");
    DFS(graph, startVertex, visited, numVertices);
    printf("\n");
    return 0;
```

Output:

```
Enter the number of vertices: 4
Enter the adjacency matrix:
0 1 1 1
1 1 0
1 0 1 0
0 0 0 1
Enter the starting vertex: 0
DFS Traversal: 0 1 2 3
```