

9 a. write a program to traverse a graph using BFS method.

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#define MAX 10
```

```
int adjMatrix[MAX][MAX];
```

```
int visited[MAX];
```

```
void bfs(int start, int n) {
```

```
    int Queue[MAX], front = 0, rear = -1;
```

```
    visited[start] = 1;
```

```
    printf("%d ", start);
```

```
    Queue[++rear] = start;
```

```
    while (front != rear) {
```

```
        int node = Queue[++front];
```

```
        for (i = 0; i < n; i++) {
```

```
            if (adjMatrix[node][i] == 1 && !visited[i])
```

```
                printf("%d ", i);
```

```
                visited[i] = 1;
```

```
                Queue[++rear] = i;
```

```
        }
```

```
    }
```

```
}
```

```
}
```



```
int main()
```

```
{
```

```
int i, n, j, start;
```

```
printf("Enter the number of vertices:");
```

```
scanf("%d", &n);
```

```
printf("Enter the adjacency matrix:");
```

```
for (i=0; i<n; i++) {
```

```
for (j=0; j<n; j++) {
```

```
scanf("%d", &adjmatrix[i][j]);
```

```
}
```

```
}
```

```
for (i=0; i<n; i++) {
```

```
visited[i] = 0;
```

```
}
```

```
printf("Enter the starting vertex:");
```

```
scanf("%d", &start);
```

```
printf("Breadth First Traversal starting from vertex (%d):", start);
```

```
bfs(start, n);
```

```
return 0;
```

```
}
```



O/P

Enter the number of vertices: 3

Enter the adjacency matrix:

0 1

1 2

2 3

3 4

4 5

Enter the starting vertex: Breadth First

Traversal starting from vertex 5: 5



23-12-24  
b. Write a program to check whether graph is connected or not using given method.

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#define MAX_VERTICES 100
```

```
typedef struct Graph {
```

```
int adj[MAX_VERTICES][MAX_VERTICES];
```

```
int vertices;
```

```
} Graph;
```

```
void DFS(Graph *g, int vertex, int visited[VERTICES]) {
```

```
for(int i=0; i<g->vertices; i++) {
```

```
if(g->adj[vertex][i] == 1 && !visited[i])
```

```
DFS(g, i, visited);
```

```
}
```

```
}
```

```
}
```

```
int isConnected(Graph *g) {
```

```
int visited[MAX_VERTICES] = {0};
```

```
DFS(g, 0, visited);
```

```
for(int i=0; i<g->vertices; i++) {
```

```
if(!visited[i])
```

```
return 0;
```

```
}
```



}

return 0;

}

int main()

{

Graph g;

int edges, u, v;

printf("Enter the number of vertices:");

scanf("%d", &g.vertices);

for(int i=0; i<g.vertices; i++)

{

for(int j=0; j<g.vertices; j++)

{

g.adj[i][j]=0;

}

printf("Enter the number of edges:");

scanf("%d", &edges);

printf("Enter the edges (u,v) where u & v are vertex indices (1 to n)");

for(int i=0; i<edges; i++)

{

scanf("%d %d", &u, &v);

g.adj[u][v]=1;

g.adj[v][u]=1;

}

if(isconnected(g))

{

printf("The graph is connected.\n");

}

printf("The graph is not connected.\n");

}

return 0;



3

return 0;

3

int main()

{

Graph g;

int edges, u, v;

printf("Enter the number of vertices:");

scanf("%d", &g.vertices);

for(int i=0; i<g.vertices; i++)

{

for(int j=0; j<g.vertices; j++)

g.adj[i][j]=0;

}

}

printf("Enter the number of edges:");

scanf("%d", &edges);

printf("Enter the edges (u,v) where u & v are vertex indices:");

for(int i=0; i<edges; i++)

scanf("%d%d", &u, &v);

g.adj[u][v]=1;

g.adj[v][u]=1;

}

if(isConnected(g))

printf("The graph is connected.\n");

}

else

printf("The graph is not connected.\n");

}

return 0;



23-12-24

output

Enter the number of vertices: 5

Enter the number of edges: 4

Enter the edges (u, v) where u and v are  
vertices indices

0 1

1 2

2 3

3 4

The graph is connected

24/12