Program

```
#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX 100

void bfs(int adj[MAX][MAX], int V, int s) {

  int q[MAX], front = 0, rear = 0;

  bool bfs_visited[MAX] = {false};

  bfs_visited[s] = true;
  q[rear++] = s;

  while (front < rear) {

    int curr = q[front++];
    printf("%d ", curr);

    for (int i = 0; i < V; i++) {
      if (adj[curr][i] == 1 && !bfs_visited[i]) {
        bfs_visited[i] = true;
        q[rear++] = i;
      }

    }
  }
}

void dfs(int adj[MAX][MAX], int V, int s, bool dfs_visited[]) {

  dfs_visited[s] = true;

  printf("%d ", s);

  for (int i = 0; i < V; i++) {
    if (adj[s][i] == 1 && !dfs_visited[i]) {
      dfs(adj, V, i, dfs_visited);
    }
  }
}

bool addEdge(int adj[MAX][MAX], bool vertices[], int u, int v) {
  adj[u][v] = 1;
  adj[v][u] = 1;
  bool newVertexAdded = false;
  if (!vertices[u]) {
    vertices[u] = true;
    newVertexAdded = true;
  }
  if (!vertices[v]) {
    vertices[v] = true;
    newVertexAdded = true;
  }
  return newVertexAdded;
}

void main() {
```

Experiment No: 28
Date:

BREADTH FIRST SEARCH AND DEPTH FIRST SEARCH

Aim:

To return the breadth first search and depth first search of a graph.

Algorithm:

1. Start
2. Declare MAX as 100
3. Define a function bfs(int adj[MAX][MAX], int V, int s)

```
initialize q[MAX], front=0, rear=0
bool bfs_visited[MAX] = {false}
bfs_visited[s]=true
q[rear++]=s
while front < rear do:
    curr = q[front++]
    print curr
    for i=0 to V do:
        if adj[curr][i] == 1 AND !bfs_visited[i]:
            bfs_visited[i] = true
            q[rear++] = i
```

4. Define a function dfs(int adj[MAX][MAX], int V, int s, bool dfs_visited[])

```
dfs_visited[s]=true
print vertex s
for i from 0 to V do:
    if adj[s][i] == 1 AND !dfs_visited[i]:
        dfs(adj,V,i,dfs_visited)
```

5. Define a function addEdge(int adj[MAX][MAX], bool vertices[], int u, int v)

```
adj[u][v] = 1
adj[v][u] = 1
newVertexAdded = false
if !vertices[u]:
    vertices[u] = true
```

```c
  int v1, v2, vs, c;
  int V = 0;
  char ch;
  int adj[MAX][MAX] = {0};
  bool vertices[MAX] = {false};
  bool visited[MAX] = {false};
  do {
    printf("\n******\n");
    printf("Graph Searching Solutions \n");
    printf("Here are your choices: \n");
    printf("1. Create a graph \n");
    printf("2. Breadth First Search \n");
    printf("3. Depth First Search \n");
    printf("Enter your choice \n");
    scanf("%d", & c);
    switch (c) {
    case 1:
      printf("Enter each edge in the form {vertex 1 vertex 2}, enter -1 -1 for edge to stop \n ");
      while (true) {
        scanf("%d%d", & v1, & v2);
        if (v1 == -1 && v2 == -1) break;
        if (addEdge(adj, vertices, v1, v2)) {
          if (v1 >= V) V++;
          if (v2 >= V) V++;
        }
      }
      printf("Edges added successfully\n");
      break;
    case 2:
      if (V > 0) {
        printf("Which vertex do you want to start from? \n");
        scanf("%d", & vs);
        printf("The breadth first search for the graph is \n");
        bfs(adj, V, vs);
        printf("\n");
      } else {
        printf("No edges have been added yet \n");
      }
      break;
    case 3:
      if (V > 0) {
        printf("Which vertex do you want to start from? \n");
        scanf("%d", & vs);
        printf("The depth first search for the graph is \n");
        dfs(adj, V, vs, visited);
        printf("\n");
      } else {
        printf("No edges have been added yet \n");
      }
      break;
    default:
      printf("Invalid choice...\n");
      break;
    }
    printf("Enter y/Y to continue \n");
    scanf(" %c", & ch);
  } while (ch == 'y' || ch == 'Y');

}
```

```
    newVertexAdded = true
if !vertices[v]:
    vertices[v] = true
    newVertexAdded = true
return newVertexAdded
```

6. Define main() function

```
initialize vertices[MAX]={false}, visited[MAX]={false}, V=0
while true
    display the operations "1.Create a Graph  2.Breadth First Search 3.Depth First Search"
    read the choice from the user
    create a switch case for the choice
        if case=1:
            while(true)
                read v1 ,v2
                if(v1 == -1&&v2 == -1) break
                if(addEdge(adj,vertices,v1,v2))
                    if(v1>=V) V++
                    if(v2>=V) V++
            break
        if case=2:
            if(V>0):
                read the vertex to start from as vs
                call bfs(adj,V,vs)
            else:
             print"No edges have been added yet "
            break
        if case=3:
            if(V>0):
                read the vertex to start from as vs
                call dfs(adj,V,vs,visited)
            else:
             print "No edges have been added yet "
            break
        else:
            print "invalid choice"
            break
    ask user choice whether to continue or not as ch
    if ch!=y OR Y:
        break
```

7. Stop

Output

```
******
Graph Searching Solutions
Here are your choices:
1. Create a graph
2. Breadth First Search
3. Depth First Search
Enter your choice
1
Enter each edge in the form {vertex 1 vertex 2}, enter -1 -1 for edge to stop
 0
1
0
2
0
3
1
4
1
5
2
5
-1
-1
Edges added successfully
Enter y/Y to continue
y

******
Graph Searching Solutions
Here are your choices:
1. Create a graph
2. Breadth First Search
3. Depth First Search
Enter your choice
2
Which vertex do you want to start from?
0
The breadth first search for the graph is
0 1 2 3 4 5
Enter y/Y to continue
y

******
Graph Searching Solutions
Here are your choices:
1. Create a graph
2. Breadth First Search
3. Depth First Search
Enter your choice
3
Which vertex do you want to start from?
0
The depth first search for the graph is
0 1 4 5 2 3
Enter y/Y to continue
n
```

Result:

Program has been executed successfully and obtained the output.