ASSIGNMENT NO:1 **DATE:29/01/2016**

PROGRAM TITLE:Traverse a Graph using Breadth First Search.

PROGRAM ALGORITHM:

```
BFS(G,v)
{
      //input:Graph 'G' represented either by adjacency matrix or adjacency list, starting vertex 'v'
      //output:printing the vertices in BFS order
      enqueue(Q,v);//Q is an empty Queue
      mark[v]=TRUE;
      while Q is not empty do
      {
            r=dequeue(Q);
            print(r);
            for each adjacent vertex 'u' of 'r' do
            {
                  if mark[u]==FALSE
                  {
                        mark[u]=TRUE;
                        enqueue(Q,u);
                  }
            }
      }
}

main()
{
      print(starting vertex);
      for each vertex u
            mark[u]=FALSE;
      BFS(G,v);
      for all unmarked vertices
            BFS(G,w);
}
```

PROGRAM CODE:

```
//C Code to Traverse a given graph using BFS
#include <stdio.h>
#include <stdlib.h>
#define MAX 50
int q[MAX];
int front=0,rear=0;
void create_graph(int **graph,int n)
{
      int i,j,x;
      for(i=0;i<n;i++)
      {
            for(j=0;j<n;j++)
            {
                  graph[i][j]=0;
            }
      }
```

```
	printf("\tIdentify the adjoining vertices:");
	for(i=0;i<n;i++)
	{
		printf("\n\tEnter the adjoining vertices of %d.Enter -99 to
stop::",i);
		for(j=0;j<n;j++)
		{
			scanf("%d",&x);
			if(x==-99)
				break;
			else
				graph[i][x]=1;
		}
	}
}
void print_graph(int **graph,int n)
{
	int i,j;
	printf("\n\tThe Adjacency Matrix of the Graph is::");
	for(i=-1;i<n;i++)
	{
		printf("\n");
		for(j=-1;j<n;j++)
		{
			if(i==-1)
				printf("%d",j);
			else if(j==-1)
				printf("%d",i);
			else
				printf("%d",graph[i][j]);
			printf("\t");
		}
	}
	printf("\n");
}
int is_full()
{
	if((rear+1)%MAX==front)
		return 1;
	else
		return 0;
}
int is_empty()
{
	if(front==rear)
		return 1;
	else
		return 0;
}
void enqueue(int n)
{
	if(is_full()==0)
	{
		q[rear]=n;
		rear=(rear+1)%MAX;
	}
	else
```

```
            printf("\tThe Queue is Full.");
}
int dequeue()
{
      if(is_empty()==0)
      {
            int x=q[front];
            front=(front+1)%MAX;
            return x;
      }
      else
            printf("\tThe Queue is Empty.");
      return -9999;
}
int bfs(int **graph,int n,int *mark,int v)
{
      int r,i;
      front=rear=0;
      enqueue(v);
      mark[v]=1;
      enqueue(-99);
      while(is_empty()==0)
      {
            r=dequeue();
            if(r!=-9999)
            {
                  if(r==-99)
                        printf("\n");
                  else
                  {
                        printf("\t%d",r);
                        for(i=0;i<n;i++)
                        {
                              if(graph[r][i]==1)
                              {
                                    if(mark[i]==0)
                                    {
                                          mark[i]=1;
                                          enqueue(i);
                                    }
                              }
                        }
                        enqueue(-99);
                  }
            }
      }
      return 0;
}
int main()
{
      int n,i,v;
      printf("\n\tEnter the number of vertices::");
      scanf("%d",&n);
      int *mark=(int*)malloc(n*sizeof(int));
      for(i=0;i<n;i++)
            mark[i]=0;
      int **graph= (int**)malloc(n*sizeof(int*));
```

```
	for(i=0;i<n;i++)
	{
		graph[i]=(int*)malloc(n*sizeof(int));
	}
	create_graph(graph,n);
	print_graph(graph,n);
	do
	{
		printf("\n\tEnter the starting vertex. Enter -99 to Quit::");
		scanf("%d",&v);
		for(i=0;i<n;i++)
			mark[i]=0;
		if(v!=-99)
		{
			printf("\tThe BFS traversal is::\n");
			bfs(graph,n,mark,v);
			for(i=0;i<n;i++)
			{
				if(mark[i]==0)
					bfs(graph,n,mark,i);
			}
		}
	}
	while(v!=-99);
	return 0;
}
```

OUTPUT:

```
	Enter the number of vertices::5
	Identify the adjoining vertices:
	Enter the adjoining vertices of 0.Enter -99 to stop::1 2 3 -99

	Enter the adjoining vertices of 1.Enter -99 to stop::0 4 3 -99

	Enter the adjoining vertices of 2.Enter -99 to stop::0 4 3 -99

	Enter the adjoining vertices of 3.Enter -99 to stop::0 1 2 4 -99

	Enter the adjoining vertices of 4.Enter -99 to stop::1 2 3 -99

	The Adjacency Matrix of the Graph is::
-1	0	1	2	3	4
0	0	1	1	1	0
1	1	0	0	1	1
2	1	0	0	1	1
3	1	1	1	0	1
4	0	1	1	1	0

	Enter the starting vertex. Enter -99 to Quit::0
	The BFS traversal is::
	0
	1	2	3
	4
```

```
Enter the starting vertex. Enter -99 to Quit::1
The BFS traversal is::
1
0    3    4
2



Enter the starting vertex. Enter -99 to Quit::2
The BFS traversal is::
2
0    3    4
1



Enter the starting vertex. Enter -99 to Quit::3
The BFS traversal is::
3
0    1    2    4




Enter the starting vertex. Enter -99 to Quit::4
The BFS traversal is::
4
1    2    3
0



Enter the starting vertex. Enter -99 to Quit::-99
```

DISCUSSION:

1. For Breadth First Search, principle of Queue is used, therefore, we had to design some more functions to maintain and perform operations on the queue.

2. We create an Adjacency Matrix for storing the graph in.