PROGRAM 1

Write program to do the following:
a. Print all the nodes reachable from a given starting node in a digraph using BFS method.

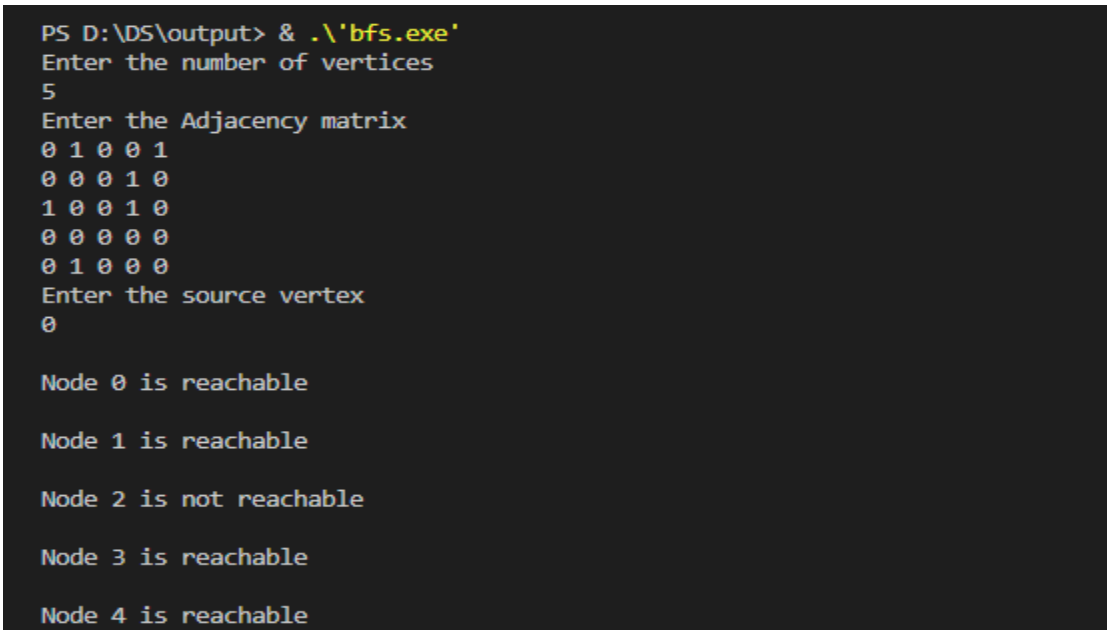
b. Check whether a given graph is connected or not using DFS method.

a.BFS

```
# include<stdio.h>
# define SIZE 20
int n, Adj[SIZE][SIZE] , i , j ,visited[SIZE];
void main()
{
   int source;
    printf("Enter the number of vertices\n");
    scanf("%d",&n);
    printf("Enter the Adjacency matrix\n");
    for(i=0;i<n;i++)
       for(j=0;j<n;j++)
          scanf("%d",&Adj[i][j]);
   for(i=0;i<n;i++)
       visited[i]=0;
   printf("Enter the source vertex\n");
   scanf("%d",&source);
   visited[source]=1;
   BFS(source);
   for(i=0;i<n;i++)
   {
       if(visited[i]!=0)
          printf("\nNode %d is reachable\n",i);
       else
          printf("\nNode %d is not reachable\n",i);
   }
}
```

```
void BFS(int source)
{
   int queue[SIZE], front, rear , u ,v;
   front=0;
   rear=-1;
   queue[++rear]=source ;
   while(front<=rear)
   {
      u=queue[front++];
      for(v=0;v<n;v++)
      {
         if(Adj[u][v]==1 && visited[v]==0)
         {
            queue[++rear] =v;
            visited[v]=1;
         }
      }
   }
}
```

OUTPUT:



OBSERVATION:

Breadth First Search

BFS

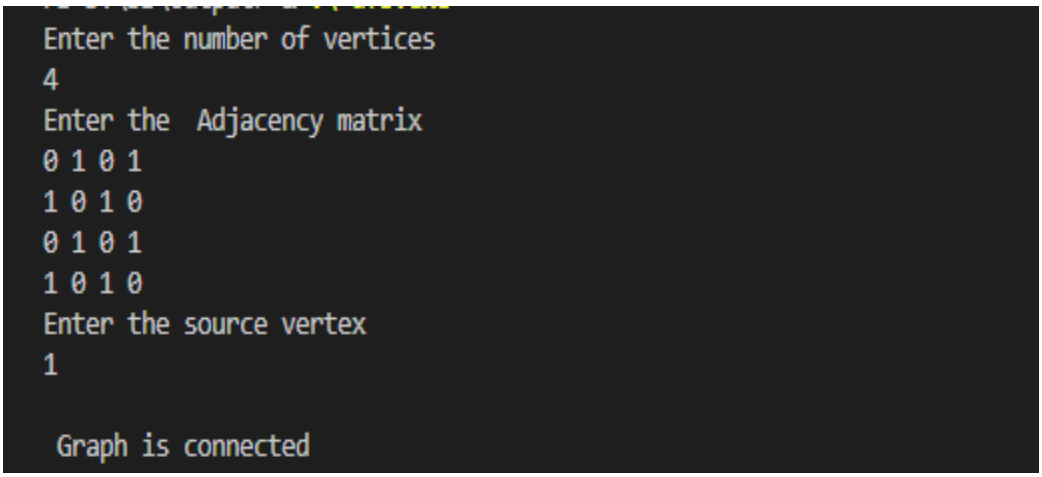
```
#include<stdio.h>
int n, i, j, visited[20], queue[10], front=-1, rear=-1;
int adj[20][20];

void bfs(int v)
{
    for(i=1; i<=n; i++)
        if(!visited[i] && adj[v][i])
            queue[++rear] = i;
        if(front<=rear)
        {
            visited[queue[front]]=1;
            bfs(queue[front++]);
        }
}
void main()
{
    int v;
    printf("Enter the number of vertices:");
    scanf("%d", &n);
    for(i=1; i<=n; i++)
    {
        queue[i]=0;
        visited[i]=0;
    }
    printf("Enter the adjacency matrix of a graph:\n");
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
            scanf("%d", &adj[i][j]);
    printf("Enter the source vertex:");
    scanf("%d", &v);
```

b. DFS

```
# include<stdio.h>
# define SIZE 20
int n,Adj[SIZE][SIZE],i,j,visited[SIZE],source;
void DFS(int s)
{
   int v;
   visited[s]=1;
   for(v=0;v<n;v++)
   {
      if(Adj[s][v]==1 && visited[v]==0)
      DFS (v);
    }
}
void main()
{
   printf("Enter the number of vertices\n");
   scanf("%d",&n);
   printf("Enter the  Adjacency matrix\n");
   for(i=0;i<n;i++)
   for(j=0;j<n;j++)
   scanf("%d",&Adj[i][j]);
   for(i=0;i<n;i++)
   visited[i]=0;
   printf("Enter the source vertex\n");
   scanf("%d",&source);
   DFS(source);
   for(i=0;i<n;i++)
   {
      if(visited[i]==0)
      {
         printf("\nGraph is not connected\n");
      }
   }
```

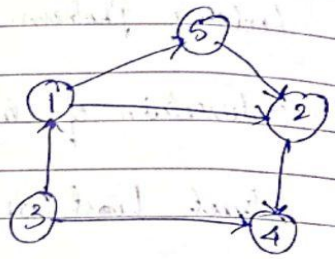
```
    printf("\n Graph is connected\n");
}
```

OUTPUT:



OBSERVATION:

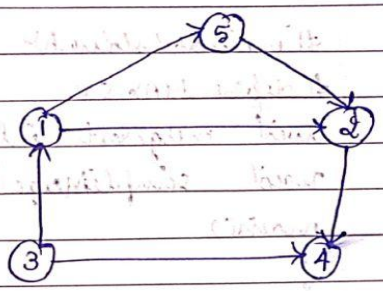
Depth First Search



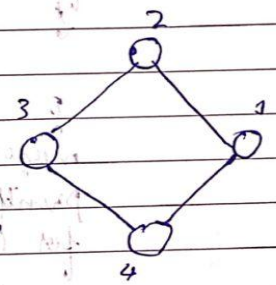
DFS

```
#include<stdio.h>
int Add[10][10], visited[10], n, i, j, source;
void DFS(int i)
{
    int j;
    printf("\n %d", i);
    visited[i] = 1;
    for(j=0; j<n; j++)
        if(!visited[j] && Add[i][j] == 1)
            DFS(j);
}
void main()
{
    printf("Enter the number of vertices: \n");
    scanf("%d", &n);
    printf("Enter adjacency matrix of the graph:");
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            scanf("%d", &Add[i][j]);
    for(i=0; i<n; i++)
        visited[i] = 0;
    printf("Enter the source node: \n");
    scanf("%d", &source);
    DFS(source);
    for(i=0; i<n; i++)
    {
        if(visited[i] == 0)
            printf("\n Graph is not connected \n");
    }
    printf("\n Graph is connected \n");
}
```

DFS output:-



Enter the number of vertices
5
Enter the adjacency matrix
0 1 0 0 1
0 0 0 1 0
1 0 0 1 0
0 0 0 0 0
0 1 0 0 0
Enter the source vertex
1
Graph is not connected.



Enter the number of vertices
4
Enter the adjacency matrix
0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 0
Enter the source vertex
1
Graph is connected.

S.T
15/6/23