```
/*****************************************************************************
Subject: DSA Laboratory
Practical No: 06
Title: A C++ Program to Represent Graph Data Structure using Adjacency Matrix and List.
      Input : A Graph (04 Nodes and 06 Edges)
      Outputs:
         a) Represent Graph using Adjacency Matrix
         b) Represent Graph using Adjacency List
         c) DFS Traversal on Adjacency Matrix Representation
         d) BFS Traversal on Adjacency List Representation


*****************************************************************************/
                        //......Header Files
#include <iostream>
using namespace std;

int adjMtx[4][4];   //....for Adjacency Matrix
int Row = 4;
int Col = 4;

struct Node         //....for Adjancency List
{
   char data;
   struct Node *down, *next;
}*Head;

                        //......Function to return Vertex Name
char vertex(int val)
{
   if(val == 0)
      return 'A';
   else if(val == 1)
      return 'B';
   else if(val == 2)
      return 'C';
   else
      return 'D';
}

                        //......Function to Create Adjacency Matrix
void create_adjMtx()
{
   int i, j;

   for(i=0; i<Row; i++)
   {
      for(j=0; j<Col; j++)
      {
```

```
            cout<<"\n Is Edge from vertex "<<vertex(i)<<" to "<<vertex(j)<<" ? : ";
            cin>>adjMtx[i][j];
        }
    }
}
```

//.....Function to Display Adjacency Matrix

```
void display_adjMtx()
{
    int i, j;

    for(i=0; i<Row; i++)
    {
        cout<<"\n";
        for(j=0; j<Col; j++)
        {
            cout<<"\t"<<adjMtx[i][j];
        }
    }
}
```

//.....Function to Create Adjacency List

```
void create_adjList()
{
    struct Node *Newnode, *move, *p;
    int i;
    int nodes;
    int edges;

    cout<<"\n\n How many Vertices in Graph: ";
    cin>>nodes;

    for(i=0; i<nodes; i++)
    {
        Newnode = new struct Node;

        Newnode->data = vertex(i);
        Newnode->down = NULL;
        Newnode->next = NULL;

        if(Head == NULL)
        {
          Head = Newnode;
          move = Head;
        }
        else
        {
```

```
            move->down = Newnode;
            move = move->down;
        }

    }

    move = Head;
    p = Head;
    while(move != NULL)
    {
        cout<<"\n How many adjacent vertices for "<<move->data<<" : ";
        cin>>edges;

        for(i=0; i<edges; i++)
        {
            Newnode = new struct Node;

            cout<<"\n\t Enter An Adjacent Vertex: ";
            cin>>Newnode->data;
            Newnode->down = NULL;
            Newnode->next = NULL;

            p->next = Newnode;
            p = p->next;
        }

        move = move->down;
          p = move;
    }

}
```

//.....Function to Display Adjacency List

```
void display_adjList()
{
    struct Node *move, *p;

    move = Head;

    while(move != NULL)
    {
        cout<<"\n\t | "<<move->data<<" |--> ";
        p = move->next;
        while(p != NULL)
        {
            cout<<p->data<<" --> ";
            p = p->next;
        }
```

```
        cout<<"NULL";

        move = move->down;
          p = move;
    }
}
```

//.....Main Function

```
int main()
{
    cout<<"\n\n A C++ Program to Represent Graph Data Structure using Adjacency Matrix
and List.";

    cout<<"\n\n 1. Creating Adjacency Matrix.....";
    create_adjMtx();

    cout<<"\n\n 2. Display Adjacency Matrix.....";
    display_adjMtx();

    cout<<"\n\n 3. Create Adjacency List.....";
    Head = NULL;
    create_adjList();

    cout<<"\n\n 4. Display Adjacency List.....";
    display_adjList();

    //DFS_Traversal();

    //BFS_Traversal();

    return 0;
}
```

/*----------OUTPUT----------

A C++ Program to Represent Graph Data Structure using Adjacency Matrix and List.

1. Creating Adjacency Matrix.....

Is Edge from vertex A to A ? : 0

Is Edge from vertex A to B ? : 1

Is Edge from vertex A to C ? : 1

Is Edge from vertex A to D ? : 1

Is Edge from vertex B to A ? : 1

Is Edge from vertex B to B ? : 0

Is Edge from vertex B to C ? : 1

Is Edge from vertex B to D ? : 1

Is Edge from vertex C to A ? : 1

Is Edge from vertex C to B ? : 1

Is Edge from vertex C to C ? : 0

Is Edge from vertex C to D ? : 1

Is Edge from vertex D to A ? : 1

Is Edge from vertex D to B ? : 1

Is Edge from vertex D to C ? : 1

Is Edge from vertex D to D ? : 0

2. Display Adjacency Matrix.....

0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

3. Create Adjacency List.....

How many Vertices in Graph: 4

How many adjacent vertices for A : 3

Enter An Adjacent Vertex: B

Enter An Adjacent Vertex: C

Enter An Adjacent Vertex: D

How many adjacent vertices for B : 3

Enter An Adjacent Vertex: A

Enter An Adjacent Vertex: C

Enter An Adjacent Vertex: D

How many adjacent vertices for C : 3

Enter An Adjacent Vertex: A

Enter An Adjacent Vertex: B

Enter An Adjacent Vertex: D

How many adjacent vertices for D : 3

Enter An Adjacent Vertex: A

Enter An Adjacent Vertex: B

Enter An Adjacent Vertex: C

4. Display Adjacency List.....

| A |--> B --> C --> D --> NULL
| B |--> A --> C --> D --> NULL
| C |--> A --> B --> D --> NULL
| D |--> A --> B --> C --> NULL

...Program finished with exit code 0
Press ENTER to exit console.

*/