//============================================================================

// Name : Graph\_BFS\_DFS.cpp

// Author : Samadhan

// Version :

// Copyright : Your copyright notice

// Description : ......DSA Lab PR-06: Graph........

//============================================================================

**#include** <iostream>

**#include** <string.h>

**using** **namespace** std;

**class** DLLNode

{

**public**:

**char** data[10];

DLLNode \*down;

DLLNode \*next;

}\*Head;

**class** Graph : **public** DLLNode

{

**public**:

DLLNode\* **create\_Graph**(DLLNode \*head);

**void** **display\_Graph**();

**void** **traverse\_DFS**();

**void** **traverse\_BFS**();

**void** **is\_Connected**();

**void** **check\_degree**();

};

DLLNode\* **Graph :: create\_Graph**(DLLNode \*head)

{

**int** i, j, vertices, adjacent;

DLLNode \*tmp, \*move, \*Newnode;

cout<<"\n\t How many vertices..? : ";

cin>>vertices;

**for**(i=0; i<vertices; i++)

{

Newnode = **new** DLLNode;

cout<<"\n\t Enter name of Vertex: ";

cin>>Newnode->data;

Newnode->next = NULL;

Newnode->down = NULL;

**if**(head == NULL)

{

head = Newnode;

move = tmp = head;

}

**else**

{

move->down = Newnode;

move = move->down;

tmp = move;

}

cout<<"\n\t How many adjacent vertices.? : ";

cin>>adjacent;

**for**(j=0; j<adjacent; j++)

{

Newnode = **new** DLLNode;

cout<<"\n\t\t Enter name of Adjacent Vertex: ";

cin>>Newnode->data;

Newnode->next = NULL;

Newnode->down = NULL;

tmp->next = Newnode;

tmp = Newnode;

}

}

**return** head;

}

**void** **Graph :: display\_Graph**()

{

DLLNode \*tmp, \*move;

cout<<"\n\n Vertex :--> Adjacent Vertices....";

move = Head;

**while**(move != NULL)

{

cout<<"\n"<<move->data<<" : ";

tmp = move->next;

**while**(tmp != NULL)

{

cout<<"-->"<<tmp->data;

tmp = tmp->next;

}

move = move->down;

}

}

**void** **Graph :: traverse\_DFS**()

{

}

**void** **Graph :: traverse\_BFS**()

{

}

**void** **Graph :: is\_Connected**()

{

}

**void** **Graph :: check\_degree**()

{

}

**int** **main**()

{

Graph G1;

Head = NULL;

cout << "\n .......DSA Lab PR-06: Graph........ ";

cout << "\n\n Operation 01: Create Graph..........";

Head = G1.create\_Graph(Head);

cout << "\n\n Operation 02: Display Graph..........";

G1.display\_Graph();

cout << "\n\n Operation 03: DFS Traversal of Graph..........";

cout << "\n\n Operation 04: BFS Traversal of Graph..........";

cout << "\n\n Operation 05: Is the Graph Connected???.......";

cout << "\n\n Operation 06: Degrees of Vertices in the Graph..........";

**return** 0;

}

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

\*

\*

.......DSA Lab PR-06: Graph........

Operation 01: Create Graph..........

How many vertices..? : 5

Enter name of Vertex: PICT

How many adjacent vertices.? : 3

Enter name of Adjacent Vertex: BHARATI

Enter name of Adjacent Vertex: KATRAJ

Enter name of Adjacent Vertex: SKNCOE

Enter name of Vertex: BHARATI

How many adjacent vertices.? : 2

Enter name of Adjacent Vertex: PICT

Enter name of Adjacent Vertex: KATRAJ

Enter name of Vertex: KATRAJ

How many adjacent vertices.? : 3

Enter name of Adjacent Vertex: PICT

Enter name of Adjacent Vertex: SKNCOE

Enter name of Adjacent Vertex: TEMPLE

Enter name of Vertex: SKNCOE

How many adjacent vertices.? : 3

Enter name of Adjacent Vertex: PICT

Enter name of Adjacent Vertex: SKNCOE

Enter name of Adjacent Vertex: TEMPLE

Enter name of Vertex: TEMPLE

How many adjacent vertices.? : 2

Enter name of Adjacent Vertex: KATRAJ

Enter name of Adjacent Vertex: SKNCOE

Operation 02: Display Graph..........

Vertex :--> Adjacent Vertices....

PICT : -->BHARATI-->KATRAJ-->SKNCOE

BHARATI : -->PICT-->KATRAJ

KATRAJ : -->PICT-->SKNCOE-->TEMPLE

SKNCOE : -->PICT-->SKNCOE-->TEMPLE

TEMPLE : -->KATRAJ-->SKNCOE

Operation 03: DFS Traversal of Graph..........

Operation 04: BFS Traversal of Graph..........

Operation 05: Is the Graph Connected???.......

Operation 06: Degrees of Vertices in the Graph..........

\*/