// ConsoleApplication11.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include <cstdlib>

using namespace std;

/\*

\* Adjacency List Node

\*/

struct AdjListNode

{

int data;

struct AdjListNode\* next;

};

/\*

\* Adjacency List

\*/

struct AdjList

{

struct AdjListNode \*head;

};

/\*

\* Class Graph

\*/

class Graph

{

private:

int V;

AdjList\* array;

public:

Graph(int V)

{

this->V = V;

array = new AdjList[V]; //total vertices

for (int i = 0; i < V; ++i)

array[i].head = NULL; //linking head of all vertices (array) to NULL ,it doesn't store any number only stores HEAD

}

/\*

\* Adding Edge to Graph

\*/

void addEdge(int src, int dest)

{

// 0-->2

// 1-->NULL

// 2-->0

// Add an edge from src to dest. A new node is added to the adjacency

// list of src. The node is added at the begining

AdjListNode\* newNode = new AdjListNode; //newNode stores both data(dest) and \*next pointer

newNode->data = dest; //consider src = 0 and dest = 1 0<----->1 for undirected graph

newNode->next = NULL; // 1----->NULL

//adding nodes at beginning of each list just like in linked list//

newNode->next = array[src].head; //\*next(of dst) storing address of head->next node i.e.. 1--->2 (first node from head)

array[src].head = newNode; // 0-->1-->2

// Since graph is undirected, add an edge from dest to src also

newNode = new AdjListNode; //now newNode storing data(src)

newNode->data = src;

newNode->next = NULL; // 0--->NULL

newNode->next = array[dest].head; // 0---->NULL (bcuz.. 1-->NULL)

array[dest].head = newNode; // 1---->0

}

/\*

\* Print the graph

\*/

void printGraph()

{

int v;

for (v = 0; v < V; ++v)

{

AdjListNode\* tmp = array[v].head; //tmp has the address of (0,1..)vertex head

cout << "\n Adjacency list of vertex " << v << "\n head ";

while (tmp)

{

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

tmp = tmp->next;

}

cout << endl;

}

}

};

/\*

\* Main

\*/

int main()

{

Graph gh(5);

gh.addEdge(0, 1);

gh.addEdge(0, 4);

gh.addEdge(1, 2);

gh.addEdge(1, 3);

gh.addEdge(1, 4);

gh.addEdge(2, 3);

gh.addEdge(3, 4);

// print the adjacency list representation of the above graph

gh.printGraph();

return 0;

}