**Experiment No.6**

**Aim:** Implementation of addition and deletion of edges in a directed graph using adjacency List.

**Code:**

/\* Adjacency list representation of a graph in C \*/

#include <stdio.h>

#include <stdlib.h>

/\* structure to represent a node of adjacency list \*/

**struct** AdjNode {

**int** dest;

**struct** AdjNode\* next;

};

/\* structure to represent an adjacency list \*/

**struct** AdjList {

**struct** AdjNode\* head;

};

/\* structure to represent the graph \*/

**struct** Graph {

**int** V; /\*number of vertices in the graph\*/

**struct** AdjList\* array;

};

**struct** AdjNode\* newAdjNode(**int** dest)

{

**struct** AdjNode\* newNode = (**struct** AdjNode\*)malloc(**sizeof**(**struct** AdjNode));

    newNode->dest = dest;

    newNode->next = NULL;

**return** newNode;

}

**struct** Graph\* createGraph(**int** V)

{

**struct** Graph\* graph = (**struct** Graph\*)malloc(**sizeof**(**struct** Graph));

    graph->V = V;

    graph->array = (**struct** AdjList\*)malloc(V \* **sizeof**(**struct** AdjList));

    /\* Initialize each adjacency list as empty by making head as NULL \*/

**int** i;

**for** (i = 0; i < V; ++i)

        graph->array[i].head = NULL;

**return** graph;

}

/\* function to add an edge to an undirected graph \*/

**void** addEdge(**struct** Graph\* graph, **int** src, **int** dest)

{

    /\* Add an edge from src to dest. The node is added at the beginning \*/

**struct** AdjNode\* check = NULL;

**struct** AdjNode\* newNode = newAdjNode(dest);

**if** (graph->array[src].head == NULL) {

        newNode->next = graph->array[src].head;

        graph->array[src].head = newNode;

}

**else** {

        check = graph->array[src].head;

**while** (check->next != NULL) {

            check = check->next;

        }

        // graph->array[src].head = newNode;

        check->next = newNode;

    }

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

    newNode = newAdjNode(src);

**if** (graph->array[dest].head == NULL) {

        newNode->next = graph->array[dest].head;

        graph->array[dest].head = newNode;

    }

**else** {

        check = graph->array[dest].head;

**while** (check->next != NULL) {

            check = check->next;

        }

        check->next = newNode;

    }

}

/\* function to print the adjacency list representation of graph\*/

**void** print(**struct** Graph\* graph)

{

**int** v;

**for** (v = 0; v < graph->V; ++v) {

**struct** AdjNode\* pCrawl = graph->array[v].head;

        printf("\n The Adjacency list of vertex %d is: \n head ", v);

**while** (pCrawl) {

            printf("-> %d", pCrawl->dest);

            pCrawl = pCrawl->next;

        }

        printf("\n");

    }

}

**int** main()

{

**int** V = 4;

**struct** Graph\* g = createGraph(V);

    addEdge(g, 0, 1);

    addEdge(g, 0, 3);

    addEdge(g, 1, 2);

    addEdge(g, 1, 3);

    addEdge(g, 2, 4);

    addEdge(g, 2, 3);

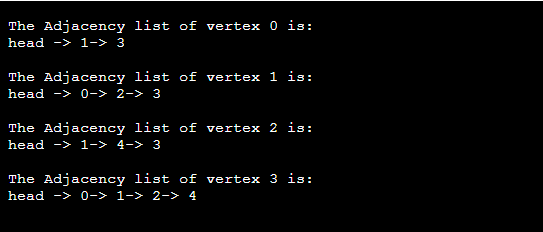
    addEdge(g, 3, 4);

    print(g);

**return** 0;

}

Output:



**Conclusion**: -Thus we have implemented addition and deletion of edges in a directed graph using adjacency list.