**LABWORK 6:**

**GROUP : CE**

**Sadikshya Pokharel Roll no: 36   
 Yural Pokhrel Roll no: 37**

**INTRODUCTION:**

In this lab work, we have implemented graph data structure.We have done following operations.

(a) isEmpty(): Returns true if the graph is empty, and false otherwise

(b) isDirected(): Returns true if the graph is directed, and false otherwise

(c) addVertex(newVertex): Inserts a new vertex to the graph

(d) addEdge(vertex1, vertex2): Adds an edge from vertex1 to vertex2

(e) removeVertex(vertexToRemove): Remove a vertex from the graph

(f) removeEdge(vertex1, vertex2): Remove an edge from the graph

(g) numVertices(): Returns the number of vertices in the graph

(h) numEdges(): Returns the number of edges in the graph

(i) indegree(vertex): Returns the indegree of a vertex

(j) outdegree(vertex): Returns the outdegree of a vertex

(k) degree(vertex): Returns the degree of a vertex

(l) neighbours(vertex): Returns the neighbours of a vertex

(m) neighbour(vertex1, vertex2): Returns true if vertex2 is a neighbour of vertex1.

**IMPLEMENTATION**:

Using adjacency matrix representation of graph, we have implemented the above operations.

An adjacency matrix of a graph G = (V, E) is a binary |V| x |V| matrix such that

aij ={1 if there exists an edge from vi to vj & 0 otherwise.

* For isEmpty() : we have checked if there are any vertices or not.
* For isDirected(): we have checked what type of graph whether directed or undirected, user has used
* For addVertex(newVertex): we have added a new row and a new column for the vertex in the matrix.
* For addEdge(vertex1, vertex2): we have set the value of av1 v2 as 1 where v1 is index of vertex1 and v2 is index of vertex2, for directed. And for undirected, we have also set the value of av2 v1 as 1 in addition.
* For removeEdge(vertex1, vertex2): we have set the value of av1 v2 as 0 where v1 is index of vertex1 and v2 is index of vertex2, for directed. And for undirected, we have also set the value of av2 v1 as 0 in addition.
* For numVertices(): we have counted the number of vertices in the graph.
* For numEdges(): we have counted the number of 1’s in the matrix and returned it for directed. And for undirected we have returned half of the counted 1’s.
* For indegree(vertex): we have counted the number of index with a[x][vertex\_index]=1, where 0<x<numofvertices
* For outdegree(vertex): we have counted the number of index with a[vertex\_index][x]=1, where 0<x<numofvertices
* For degree(vertex): we have added the indegree and outdegree of vertex and returned it for directed graph, whereas returned half of it for undirected graph.
* For neighbour(vertex1, vertex2): we have returned true if av1 v2 is 1, else false is returned.

**INPUT:**

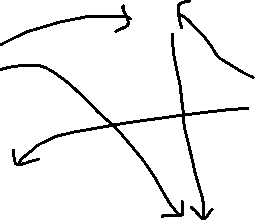
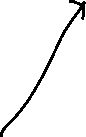
We have used the following graph in out program:

For directed:

B

A

C



E

D

For undirected:

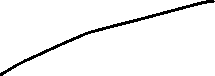
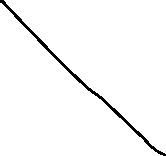
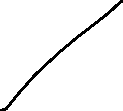
B



A



C



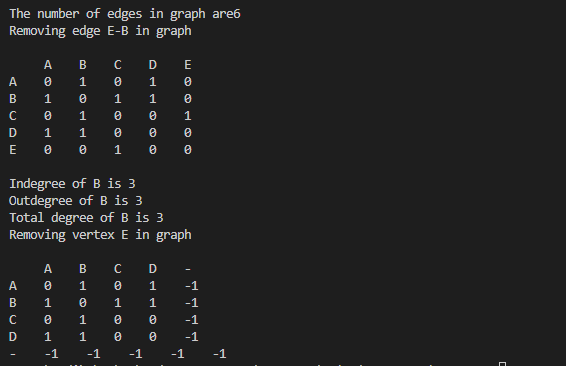
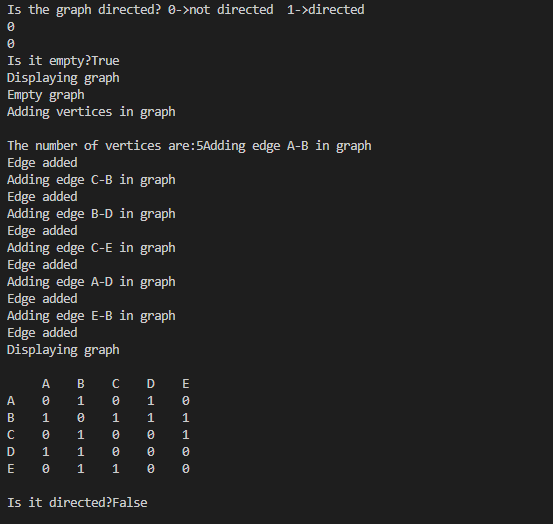
E

D

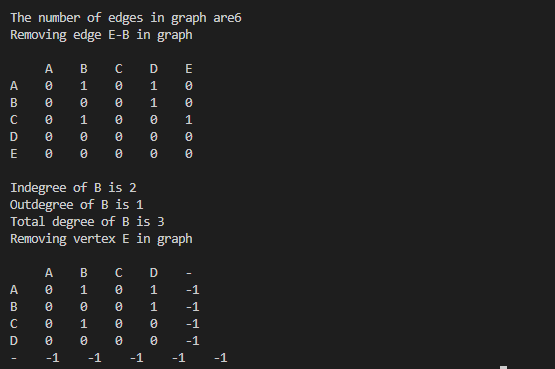
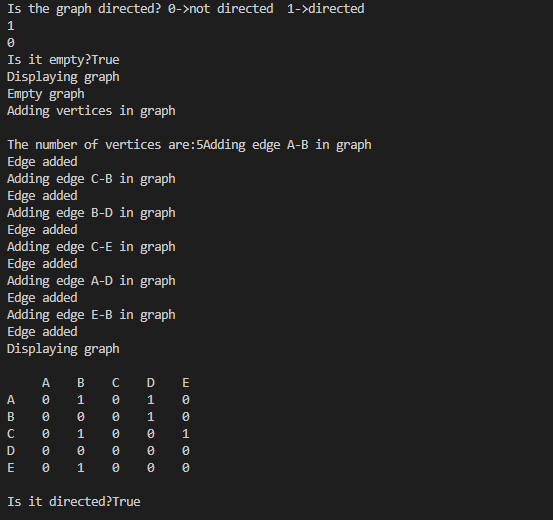
**OUTPUT:**

Below inserted are the screenshots of output of the program.

**For undirected:**



**For directed:**

****