```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>


//Membuat struct node

struct node{

   int vertex;

   struct node* next;

};


//Inisialisasi variable struct node

struct node* createNode(int v);


//Membuat struct graph

struct Graph{

   int numVertices;

   int* visited;

   struct node** adjLists;

};


//Membuat graph

struct Graph* createGraph(int vertices){

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

   graph->numVertices = vertices;

   graph->adjLists = malloc(vertices * sizeof(struct node*));

   graph->visited = malloc(vertices * sizeof(int));

   int i;

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

      graph->adjLists[i] = NULL;

      graph->visited[i] = 0;

   }
```

```
    return graph;
}

//Struct yang diinisialisasi di atas
struct node* createNode(int v){
    struct node* newNode = malloc(sizeof(struct node));
    newNode->vertex = v;
    newNode->next = NULL;
    return newNode;
}

//Fungsi menampilkan menu
void print(){
    puts("Graph Representation and Transversal");
    puts("===================================");
    puts("\t\t   0\n");
    puts("\t\t /   \\\n");
    puts("\t       1 - 4 - 2\n");
    puts("\t        \\  /   \n");
    puts("\t\t  3\n");
    puts("1. Show Adjacency Matrix");
    puts("2. Show Adjacency List");
    puts("3. Show Degree of all vertices");
    puts("4. Show BFS Traversal from vertex 0");
    puts("5. Show DFS Traversal from vertex 0");
    puts("6. Exit");
}

//Initialize
void init(int arr[][5]){
    int i,j;
    for(i = 0; i < 5; i++)
        for(j = 0; j < 5; j++)
            arr[i][j] = 0;
}

//Untuk adjacency matrix
void edgeMatrix(int arr[][5],int src, int dest){
     arr[src][dest] = 1;
}

void adjMatrix(int arr[][5]){
    puts("\n\tAdjacecncy Matrix of this Graph");
    puts("\t_________________________________\n");
    puts("   Vertex      0   1   2   3   4 \n");
    int i, j;
    for(i = 0; i < 5; i++){
        printf("          %d ",i);
        for(j = 0; j < 5; j++){
            printf("  %d ",arr[i][j]);
        }
        printf("\n\n");
    }
}

void degree(int arr[][5]){
    int i,j,ctr;
    printf("\nDegree of All Vertices\n");
    printf("----------------------\n");
    for(i=0;i<5;i++){
        ctr=0;
        for(j=0;j<5;j++){
```

```
        if(arr[i][j]==1){
          ctr++;
        }
        }
        printf("Degree of all vertex %d : %d\n\n", i, ctr);
    }
}

//Untuk adjacency list
void edgeList(struct Graph* graph, int s, int d){
    struct node* newNode = createNode(d);
    newNode->next = graph->adjLists[s];
    graph->adjLists[s] = newNode;
    newNode = createNode(s);
    newNode->next = graph->adjLists[d];
    graph->adjLists[d] = newNode;
}

void adjList(struct Graph* graph){
    int v;
    printf("\nAdjacency List of this Graph\n");
    printf("______________________________\n");
    for (v = 0; v < graph->numVertices; v++) {
        struct node* temp = graph->adjLists[v];
        printf("\n Vertex %d : ", v);
        while (temp) {
          printf("%d -> ", temp->vertex);
          temp = temp->next;
        }
        printf("NULL");
        printf("\n");
    }
}

//Fungsi utama
int main(){
    int input;
    while(1){
        print();
        printf(">> ");
        scanf("%d",&input);
        if(input == 1){
            int matrix[5][5];
            init(matrix);
            edgeMatrix(matrix,0,1);
            edgeMatrix(matrix,0,2);
            edgeMatrix(matrix,1,0);
            edgeMatrix(matrix,1,4);
            edgeMatrix(matrix,1,3);
            edgeMatrix(matrix,2,0);
            edgeMatrix(matrix,2,4);
            edgeMatrix(matrix,3,1);
            edgeMatrix(matrix,3,4);
            edgeMatrix(matrix,4,1);
            edgeMatrix(matrix,4,2);
            edgeMatrix(matrix,4,3);
            adjMatrix(matrix);
            getchar();
            printf("\n\tEnter to Continue.......");
            getchar();
            system("cls");
        }
```

```
else if(input == 2){
    struct Graph* graph = createGraph(5);
    edgeList(graph, 3, 4);
    edgeList(graph, 2, 4);
    edgeList(graph, 1, 4);
    edgeList(graph, 1, 3);
    edgeList(graph, 0, 2);
    edgeList(graph, 0, 1);
    adjList(graph);
    getchar();
    printf("\n\tEnter to Continue.......");
    getchar();
    system("cls");
}
else if(input == 3){
    int matrix[5][5];
    init(matrix);
    edgeMatrix(matrix,0,1);
    edgeMatrix(matrix,0,2);
    edgeMatrix(matrix,1,0);
    edgeMatrix(matrix,1,4);
    edgeMatrix(matrix,1,3);
    edgeMatrix(matrix,2,0);
    edgeMatrix(matrix,2,4);
    edgeMatrix(matrix,3,1);
    edgeMatrix(matrix,3,4);
    edgeMatrix(matrix,4,1);
    edgeMatrix(matrix,4,2);
    edgeMatrix(matrix,4,3);
    degree(matrix);
    getchar();
```

```
            printf("\n\tEnter to Continue.......");
            getchar();
            system("cls");
        }
        else if(input == 4){
            printf("\n--- Empty ---\n\n");
            getchar();
            printf("\n\tEnter to Continue.......");
            getchar();
            system("cls");
        }
        else if(input == 5){
            printf("\n--- Empty ---\n\n");
            getchar();
            printf("\n\tEnter to Continue.......");
            getchar();
            system("cls");
        }
        else if(input == 6){
            exit:
            break;
        }
        else{
            system("cls");
        }
    }
    return 0;
}
```