BREATH FIRST SEARCH:

```
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a graph node
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Define the structure for a graph
typedef struct Graph {
    int numVertices;
    Node** adjLists;
} Graph;

// Function to create a new graph node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to create a new graph
Graph* createGraph(int numVertices) {
    Graph* graph = (Graph*)malloc(sizeof(Graph));
    graph->numVertices = numVertices;
    graph->adjLists = (Node**)malloc(numVertices * sizeof(Node*));
    for (int i = 0; i < numVertices; i++) {
        graph->adjLists[i] = NULL;
```

```
    }
    return graph;
}

// Function to add an edge to the graph
void addEdge(Graph* graph, int src, int dest) {
    Node* newNode = createNode(dest);
    newNode->next = graph->adjLists[src];
    graph->adjLists[src] = newNode;
}

// Function to perform BFS traversal
void BFS(Graph* graph, int startVertex) {
    int visited[graph->numVertices];
    for (int i = 0; i < graph->numVertices; i++) {
        visited[i] = 0;
    }
    visited[startVertex] = 1;
    Node* queue[graph->numVertices];
    int front = 0, rear = 0;
    queue[rear++] = graph->adjLists[startVertex];
    while (front < rear) {
        Node* temp = queue[front++];
        while (temp != NULL) {
            printf("%d ", temp->data);
            visited[temp->data] = 1;
            temp = temp->next;
        }
        for (int i = 0; i < graph->numVertices; i++) {
            if (!visited[i]) {
                queue[rear++] = graph->adjLists[i];
            }
```

```
        }
    }
}

int main() {
    Graph* graph = createGraph(6);
    addEdge(graph, 0, 1);
    addEdge(graph, 0, 2);
    addEdge(graph, 1, 3);
    addEdge(graph, 1, 4);
    addEdge(graph, 2, 5);
    printf("BFS Traversal: ");
    BFS(graph, 0);
    return 0;
}
```

Output:

BFS Traversal: 0 1 2 3 4 5

DEPTH FIRST SEARCH:

```
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a graph node
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Define the structure for a graph
typedef struct Graph {
```

```
    int numVertices;
    Node** adjLists;
} Graph;

// Function to create a new graph node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to create a new graph
Graph* createGraph(int numVertices) {
    Graph* graph = (Graph*)malloc(sizeof(Graph));
    graph->numVertices = numVertices;
    graph->adjLists = (Node**)malloc(numVertices * sizeof(Node*));
    for (int i = 0; i < numVertices; i++) {
        graph->adjLists[i] = NULL;
    }
    return graph;
}

// Function to add an edge to the graph
void addEdge(Graph* graph, int src, int dest) {
    Node* newNode = createNode(dest);
    newNode->next = graph->adjLists[src];
    graph->adjLists[src] = newNode;
}

// Function to perform DFS traversal
void DFS(Graph* graph, int startVertex) {
```

```
    int visited[graph->numVertices];
    for (int i = 0; i < graph->numVertices; i++) {
        visited[i] = 0;
    }
    DFSUtil(graph, startVertex, visited);
}

void DFSUtil(Graph* graph, int vertex, int* visited) {
    visited[vertex] = 1;
    printf("%d ", vertex);
    Node* temp = graph->adjLists[vertex];
    while (temp != NULL) {
        if (!visited[temp->data]) {
            DFSUtil(graph, temp->data, visited);
        }
        temp = temp->next;
    }
}

int main() {
    Graph* graph = createGraph(6);
    addEdge(graph, 0, 1);
    addEdge(graph, 0, 2);
    addEdge(graph, 1, 3);
    addEdge(graph, 1, 4);
    addEdge(graph, 2, 5);
    printf("DFS Traversal: ");
    DFS(graph, 0);
    return 0;
}
```

Output:

DFS Traversal: 0 1 3 4 2 5