1. Write a C program for Unweighted Graph.

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int dest;
    struct Node* next;
};
struct AdjList {
    struct Node* head;
};
struct Graph {
    int V;
    struct AdjList* array;
};
struct Node* newAdjListNode(int dest) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    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));
    for (int i = 0; i < V; ++i)
        graph->array[i].head = NULL;
    return graph;
}
void addEdge(struct Graph* graph, int src, int dest) {
    struct Node* newNode = newAdjListNode(dest);
    newNode->next = graph->array[src].head;
    graph->array[src].head = newNode;
    newNode = newAdjListNode(src);
    newNode->next = graph->array[dest].head;
    graph->array[dest].head = newNode;
}
```

```
/tmp/3y23T25YOh.o

Adjacency list of vertex 0
head -> 2-> 1

Adjacency list of vertex 1
head -> 3-> 2-> 0

Adjacency list of vertex 2
head -> 3-> 1-> 0

Adjacency list of vertex 3
head -> 2-> 1


=== Code Execution Successful ===
```

```c
    newNode->next = graph->array[dest].head;
    graph->array[dest].head = newNode;
}
void printGraph(struct Graph* graph) {
    for (int v = 0; v < graph->V; ++v) {
        struct Node* pCrawl = graph->array[v].head;
        printf("\n Adjacency list of vertex %d\n head ", v);
        while (pCrawl) {
            printf("-> %d", pCrawl->dest);
            pCrawl = pCrawl->next;
        }
        printf("\n");
    }
}
int main() {
    int V = 4;
    struct Graph* graph = createGraph(V);
    addEdge(graph, 0, 1);
    addEdge(graph, 0, 2);
    addEdge(graph, 1, 2);
    addEdge(graph, 1, 3);
    addEdge(graph, 2, 3);
    printGraph(graph);
    return 0;
}
```

```
head -> 3-> 1-> 0

Adjacency list of vertex 3
head -> 2-> 1


=== Code Execution Successful ===
```

2.Write a C program for Weighted Graph.

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int dest;
    int weight;
    struct Node* next;
};
struct AdjList {
    struct Node* head;
};
struct Graph {
    int V;
    struct AdjList* array;
};
struct Node* newAdjListNode(int dest, int weight) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->dest = dest;
    newNode->weight = weight;
    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));
    for (int i = 0; i < V; ++i)
        graph->array[i].head = NULL;
    return graph;
}
void addEdge(struct Graph* graph, int src, int dest, int weight) {
    struct Node* newNode = newAdjListNode(dest, weight);
    newNode->next = graph->array[src].head;
    graph->array[src].head = newNode;
    newNode = newAdjListNode(src, weight);
    newNode->next = graph->array[dest].head;
```

```
/tmp/EyHbDmT6JT.o

Adjacency list of vertex 0
head -> 2 (weight = 3) -> 1 (weight = 2)

Adjacency list of vertex 1
head -> 3 (weight = 4) -> 2 (weight = 1) -> 0 (weight = 2)

Adjacency list of vertex 2
head -> 3 (weight = 2) -> 1 (weight = 1) -> 0 (weight = 3)

Adjacency list of vertex 3
head -> 2 (weight = 2) -> 1 (weight = 4)


=== Code Execution Successful ===
```

```c
    graph->array[src].head = newNode;
    newNode = newAdjListNode(src, weight);
    newNode->next = graph->array[dest].head;
    graph->array[dest].head = newNode;
}
void printGraph(struct Graph* graph) {
    for (int v = 0; v < graph->V; ++v) {
        struct Node* pCrawl = graph->array[v].head;
        printf("\n Adjacency list of vertex %d\n head ", v);
        while (pCrawl) {
            printf("-> %d (weight = %d) ", pCrawl->dest, pCrawl->weight);
            pCrawl = pCrawl->next;
        }
        printf("\n");
    }
}
int main() {
    int V = 4;
    struct Graph* graph = createGraph(V);
    addEdge(graph, 0, 1, 2);
    addEdge(graph, 0, 2, 3);
    addEdge(graph, 1, 2, 1);
    addEdge(graph, 1, 3, 4);
    addEdge(graph, 2, 3, 2);
    printGraph(graph);
    return 0;
}
```

```
Adjacency list of vertex 2
head -> 3 (weight = 2) -> 1 (weight = 1) -> 0 (weight = 3)

Adjacency list of vertex 3
head -> 2 (weight = 2) -> 1 (weight = 4)


=== Code Execution Successful ===
```