

1. Write a C program for Unweighted Graph.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 struct Node {
4     int dest;
5     struct Node* next;
6 };
7 struct AdjList {
8     struct Node* head;
9 };
10 struct Graph {
11     int V;
12     struct AdjList* array;
13 };
14 struct Node* newAdjListNode(int dest) {
15     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
16     newNode->dest = dest;
17     newNode->next = NULL;
18     return newNode;
19 }
20 struct Graph* createGraph(int V) {
21     struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
22     graph->V = V;
23     graph->array = (struct AdjList*)malloc(V * sizeof(struct AdjList));
24     for (int i = 0; i < V; ++i)
25         graph->array[i].head = NULL;
26     return graph;
27 }
28 void addEdge(struct Graph* graph, int src, int dest) {
29     struct Node* newNode = newAdjListNode(dest);
30     newNode->next = graph->array[src].head;
31     graph->array[src].head = newNode;
32     newNode = newAdjListNode(src);
33     newNode->next = graph->array[dest].head;
34     graph->array[dest].head = newNode;
35 }
36 void printGraph(struct Graph* graph) {
37     for (int v = 0; v < graph->V; ++v) {
38         struct Node* pCrawl = graph->array[v].head;
39         printf("\n Adjacency list of vertex %d\n head ", v);
40         while (pCrawl) {
41             printf("-> %d", pCrawl->dest);
42             pCrawl = pCrawl->next;
43         }
44         printf("\n");
45     }
46 }
47 int main() {
48     int V = 4;
49     struct Graph* graph = createGraph(V);
50     addEdge(graph, 0, 1);
51     addEdge(graph, 0, 2);
52     addEdge(graph, 1, 2);
53     addEdge(graph, 1, 3);
54     addEdge(graph, 2, 3);
55     printGraph(graph);
56     return 0;
57 }
```

```
/tmp/3y23T25Y0h.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 ===

head -> 3-> 1-> 0

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

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

2. Write a C program for Weighted Graph.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 struct Node {
4     int dest;
5     int weight;
6     struct Node* next;
7 };
8 struct AdjList {
9     struct Node* head;
10 };
11 struct Graph {
12     int V;
13     struct AdjList* array;
14 };
15 struct Node* newAdjListNode(int dest, int weight) {
16     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
17     newNode->dest = dest;
18     newNode->weight = weight;
19     newNode->next = NULL;
20     return newNode;
21 }
22 struct Graph* createGraph(int V) {
23     struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
24     graph->V = V;
25     graph->array = (struct AdjList*)malloc(V * sizeof(struct AdjList));
26     for (int i = 0; i < V; ++i)
27         graph->array[i].head = NULL;
28     return graph;
29 }
30 void addEdge(struct Graph* graph, int src, int dest, int weight) {
31     struct Node* newNode = newAdjListNode(dest, weight);
32     newNode->next = graph->array[src].head;
33     graph->array[src].head = newNode;
34     newNode = newAdjListNode(src, weight);
35     newNode->next = graph->array[dest].head;
36     graph->array[dest].head = newNode;
37 }
38 void printGraph(struct Graph* graph) {
39     for (int v = 0; v < graph->V; ++v) {
40         struct Node* pCrawl = graph->array[v].head;
41         printf("\n Adjacency list of vertex %d\n head ", v);
42         while (pCrawl) {
43             printf("-> %d (weight = %d) ", pCrawl->dest, pCrawl->weight);
44             pCrawl = pCrawl->next;
45         }
46         printf("\n");
47     }
48 }
49 int main() {
50     int V = 4;
51     struct Graph* graph = createGraph(V);
52     addEdge(graph, 0, 1, 2);
53     addEdge(graph, 0, 2, 3);
54     addEdge(graph, 1, 2, 1);
55     addEdge(graph, 1, 3, 4);
56     addEdge(graph, 2, 3, 2);
57     printGraph(graph);
58     return 0;
59 }
```

/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 ===

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 ===