Bellman-Ford Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<limits.h>
struct Edge
{
	int Source;
	int Destination;
	int Weight;
};

struct Graph
{
	int VerticesCount;
	int EdgesCount;
	struct Edge* edge;
};

struct Graph* CreateGraph(int verticesCount, int edgesCount)
{
	struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
	graph->VerticesCount = verticesCount;
	graph->EdgesCount = edgesCount;
	graph->edge = (struct Edge*)malloc(graph->EdgesCount * sizeof(struct Edge));

	return graph;
}

void Print(int distance[], int count)
{
	printf("Vertex   Distance from source\n");

	for (int i = 0; i < count; ++i)
		printf("%d\t %d\n", i, distance[i]);
}

void BellmanFord(struct Graph* graph, int source)
{
	int verticesCount = graph->VerticesCount;
	int edgesCount = graph->EdgesCount;
	int* distance = (int*)malloc(sizeof(int) * verticesCount);

	for (int i = 0; i < verticesCount; i++)
```

```
        distance[i] = INT_MAX;

    distance[source] = 0;

    for (int i = 1; i <= verticesCount - 1; ++i)
    {
        for (int j = 0; j < edgesCount; ++j)
        {
            int u = graph->edge[j].Source;
            int v = graph->edge[j].Destination;
            int weight = graph->edge[j].Weight;

            if (distance[u] != INT_MAX && distance[u] + weight < distance[v])
                distance[v] = distance[u] + weight;
        }
    }

    for (int i = 0; i < edgesCount; ++i)
    {
        int u = graph->edge[i].Source;
        int v = graph->edge[i].Destination;
        int weight = graph->edge[i].Weight;

        if (distance[u] != INT_MAX && distance[u] + weight < distance[v])
            printf("Graph contains negative weight cycle.");
    }

    Print(distance, verticesCount);
}
```

Output:

```
Vertex   Distance from source
0        0
1        -1
2        2
3        -2
4        1
```