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

#define MAX_VERTICES 100

// Structure to represent an edge
struct Edge {
    int src, dest, weight;
};

// Structure to represent a graph
struct Graph {
    int V, E;
    struct Edge* edge;
};

// Function to create a graph with V vertices and E edges
struct Graph* createGraph(int V, int E) {
    struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
    graph->V = V;
    graph->E = E;
    graph->edge = (struct Edge*)malloc(E * sizeof(struct Edge));
    return graph;
}

// Function to perform Bellman-Ford algorithm to find shortest paths from source
void bellmanFord(struct Graph* graph, int src) {
    int V = graph->V;
    int E = graph->E;
    int dist[V];
```

```
// Initialize distances from source to all other vertices as INFINITE
for (int i = 0; i < V; ++i)
    dist[i] = INT_MAX;
dist[src] = 0;

// Relax all edges |V| - 1 times
for (int i = 1; i <= V - 1; ++i) {
    for (int j = 0; j < E; ++j) {
        int u = graph->edge[j].src;
        int v = graph->edge[j].dest;
        int weight = graph->edge[j].weight;
        if (dist[u] != INT_MAX && dist[u] + weight < dist[v])
            dist[v] = dist[u] + weight;
    }
}

// Check for negative weight cycles
for (int i = 0; i < E; ++i) {
    int u = graph->edge[i].src;
    int v = graph->edge[i].dest;
    int weight = graph->edge[i].weight;
    if (dist[u] != INT_MAX && dist[u] + weight < dist[v]) {
        printf("Graph contains negative weight cycle\n");
        return;
    }
}

// Print the shortest distances
printf("Vertex   Distance from Source\n");
for (int i = 0; i < V; ++i)
```

```
        printf("%d \t\t %d\n", i, dist[i]);
}

int main() {
    int V, E;
    printf("Enter number of vertices and edges: ");
    scanf("%d %d", &V, &E);

    struct Graph* graph = createGraph(V, E);

    printf("Enter source, destination and weight for each edge:\n");
    for (int i = 0; i < E; ++i) {
        scanf("%d %d %d", &graph->edge[i].src, &graph->edge[i].dest, &graph->edge[i].weight);
    }

    int source;
    printf("Enter source vertex: ");
    scanf("%d", &source);

    bellmanFord(graph, source);

    return 0;
}
```

OUTPUT:

```
Enter number of vertices and edges: 5 8
Enter source, destination and weight for each edge:
0 1 6
0 2 7
1 2 8
1 3 5
1 4 -4
2 3 -3
2 4 9
3 1 -2
Enter source vertex: 0
Vertex   Distance from Source
0        0
1        2
2        7
3        4
4        5
```