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

struct node {
        int index;
        int weight;
        struct node *next;

};

struct node* insert(int i, int weight) {
        struct node * n = (struct node *) malloc(sizeof(struct node));
        n->index = i;
        n->weight = weight;
        n->next = NULL;
        return n;

}
struct node *vertices[10];
int d[10], p[10], s, v, e;
int flag = 0;
void relaxation(int s);
void path(int i);

//main method
int main() {
        int i;

        struct node * temp;
        temp = (struct node *) malloc(sizeof(struct node));

        printf("\n print enter no of nodes in graph:");
        scanf("%d", &v);

        for (i = 0; i < v; i++) {
                vertices[i] = insert(i, 0);
        }

        int v1, v2, weight;
        printf("\n Enter How Many edges are there in graph:");
        scanf("%d", &e);
        for (i = 0; i < e; i++) {
                printf("\n Enter Start vertex");
                scanf("%d", &v1);
                printf("\n Enter end vertex");
                scanf("%d", &v2);
                printf("\n Enter  weight");
                scanf("%d", &weight);
                temp = vertices[v1];
                printf("%d", temp->index);
                while (temp->next != NULL) {
                        temp = temp->next;
                }
                temp->next = insert(v2, weight);
```

```
        }

        for (i = 0; i < v; i++) {
                temp = vertices[i];
                printf("\n%d:", temp->index);
                while (temp != NULL) {

                        printf("->%d[%d]", temp->index, temp->weight);
                        temp = temp->next;

                }
        }

        printf("\nEnter Source Vertex:");
        scanf("%d", &s);
        relaxation(s);
        return 0;

}

void relaxation(int s) {
        int i, j;
        struct node * temp;
        temp = (struct node *) malloc(sizeof(struct node));

        for (i = 0; i < v; i++) {
                d[i] = 32767;
                p[i] = -1;
        }
        int update = 0, k;
        d[s] = 0;
        p[s] = s;

        for (i = 0; i < v; i++) {
                update = 0; //setting flag  for early exit
                for (j = 0; j < v; j++) {
                        temp = vertices[j]->next;

                        while (temp != NULL) {

                                if (d[j] != 32767 && d[temp->index] >
d[j] + temp->weight) {
                                        d[temp->index] = d[j] + temp-
>weight;
                                        p[temp->index] = j;
                                        update = 1;
                                }
                                temp = temp->next;

                        }

                }
                if (update == 0) {
                        printf("\n**early break at %d", i);
                        break;

                }
```

```
                printf("\nIteration No:%d", i); //printing each
iteration for understanding
                for (k = 0; k < v; k++) {
                        printf("\n%d|%d  ", d[k], p[k]);
                }

        }

        //detecting negative weight cycle
        for (j = 0; j < v; j++) {
                temp = vertices[j]->next;

                while (temp != NULL) {
                        if (d[j] != 32767 && d[temp->index] > d[j] +
temp->weight) {
                                printf("\n Negative weight cycle
detected");
                                flag = 1;
                                break;
                        }

                        temp = temp->next;
                }
        }

//Printing path

        if (flag == 0) //print path only if negative cycle is not
there
                        {
                for (i = 0; i < v; i++) {
                        printf("\n%d:", i);
                        if (p[i] == s) {
                                printf("->%d", s);
                        } else if (p[i] != s) {
                                printf("%d", s);
                                path(i);
                                printf("->%d", i);

                        }

                }
        }
}

void path(int i) {

        if (p[i] != s)
        {
                path(p[i]);
                printf("->%d", p[i]);
        }

}


/*
```

```
itexam@c05l0720:~$ gcc ass3.c
itexam@c05l0720:~$ ./a.out

 print enter no of nodes in graph:5

 Enter How Many edges are there in graph:6

 Enter Start vertex1

 Enter end vertex2

 Enter  weight2
1
 Enter Start vertex2

 Enter end vertex3

 Enter  weight2
2
 Enter Start vertex2

 Enter end vertex4

 Enter  weight1
2
 Enter Start vertex4

 Enter end vertex2

 Enter  weight1
4
 Enter Start vertex3

 Enter end vertex4

 Enter  weight-4
3
 Enter Start vertex4

 Enter end vertex5

 Enter  weight3
4
0:->0[0]
1:->1[0]->2[2]
2:->2[0]->3[2]->4[1]
3:->3[0]->4[-4]
4:->4[0]->2[1]->5[3]
Enter Source Vertex:1

Iteration No:0
32767|-1
0|1
1|4
4|2
0|3
Iteration No:1
```

```
32767|-1
0|1
0|4
3|2
-1|3
Iteration No:2
32767|-1
0|1
-1|4
2|2
-2|3
Iteration No:3
32767|-1
0|1
-2|4
1|2
-3|3
Iteration No:4
32767|-1
0|1
-3|4
0|2
-4|3
 Negative weight cycle detected

*/
```