```cpp
#include <algorithm>
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;

typedef pair<int, int> ii;
typedef vector<int> vi;
typedef vector<ii> vii;
#define INF 1000000000

int main() {
  int V, E, s, a, b, w;
  vector<vii> AdjList;

  /*
  // Graph in Figure 4.18, has negative weight, but no negative cycle
  5 5 0
  0 1 1
  0 2 10
  1 3 2
  2 3 -10
  3 4 3

  // Graph in Figure 4.19, negative cycle exists
  3 3 0
  0 1 1000
  1 2 15
  2 1 -42
  */

  freopen("in_06.txt", "r", stdin);

  scanf("%d %d %d", &V, &E, &s);

  AdjList.assign(V, vii()); // assign blank vectors of pair<int, int>s to AdjList
  for (int i = 0; i < E; i++) {
    scanf("%d %d %d", &a, &b, &w);
    AdjList[a].push_back(ii(b, w));
  }

  // Bellman Ford routine
  vi dist(V, INF); dist[s] = 0;
  for (int i = 0; i < V - 1; i++)  // relax all E edges V-1 times, overall O(VE)
    for (int u = 0; u < V; u++)                          // these two loops = O(E)
      for (int j = 0; j < (int)AdjList[u].size(); j++) {
        ii v = AdjList[u][j];        // we can record SP spanning here if needed
        dist[v.first] = min(dist[v.first], dist[u] + v.second);         // relax
      }

  bool hasNegativeCycle = false;
  for (int u = 0; u < V; u++)                            // one more pass to check
    for (int j = 0; j < (int)AdjList[u].size(); j++) {
      ii v = AdjList[u][j];
      if (dist[v.first] > dist[u] + v.second)                  // should be false
        hasNegativeCycle = true;     // but if true, then negative cycle exists!
    }
  printf("Negative Cycle Exist? %s\n", hasNegativeCycle ? "Yes" : "No");

  if (!hasNegativeCycle)
    for (int i = 0; i < V; i++)
      printf("SSSP(%d, %d) = %d\n", s, i, dist[i]);

  return 0;
}
```