CS39002: Operating Systems Lab.
Assignment 7
Floating date: 21/3/2016
Due date: 28/3/2016

In this assignment we will implement Floyd-Warshall All-Pairs-Shortest-Path algorithm in a multi-threaded fashion along with enforcing the readers-writers problem. The single-threaded version is given below:

Algorithm: Floyd-Warshall (FW) algorithm

```
1 for k = 1 to n do
2    for i = 1 to n do
3       for j = 1 to n do
4          if (dist[i][k] + dist[k][j] < dist[i][j])
5              dist[i][j] = dist[i][k] + dist[k][j]
6       end for
7    end for
8 end for
```

Here the outer **k** loop denotes that we are including k as an intermediate vertex in this iteration. **dist** matrix represents the distance matrix (dist[i][j] = minimum distance between i and j in the graph till the current iteration). We need to maintain **only 1 copy** of the matrix. In iteration number k, we only need to read values from the kth column and the kth row of the matrix and we can update values (i,j) in the same copy of the matrix without any inconsistency.

Multi-threaded implementation:

The k loop remains same. Here instead of the i loop, we create n threads that represent each iteration of the i loop. Each thread runs the j loop from 1 to n. There is a global matrix **Graph** that stores the adjacency matrix for the graph. All the threads can read it simultaneously. There is another global matrix **dist** that is used for updating the minimum distances between the vertices. Only 1 copy of dist will suffice.

At the start of every iteration inside the k loop, you need to create n threads. Each will run it's own j loop. You have to enforce **readers-writers problem** in the manner of how the dist matrix is accessed by the different threads. Any number of threads can read from the dist matrix provided that no other thread is writing to it. Only 1 thread can write to the dist matrix at a time. If you observe closely, line 4 has only **read statements** and line 5 is the only **write statement** present in the algorithm.

At the end of every iteration of the k loop, you need to wait for all the threads to finish. Then only can you start another iteration of the k loop. So you need to **join** the threads at the end of every iteration of the k loop.

Input Format: Graph structure: The first line contains 2 integers N and M. N is the number of nodes. M is the number of **undirected** edges. Then each entry contains the link information (M entries). Line i+1 contains 3 integers u_i, v_i and w_i which represents an undirected edge between nodes u_i and v_i and w_i is the weight of that edge.

Constraints:
1<= N <= 100
All edge weights are positive.

Output Format: Print the final dist matrix. In case node j is not reachable from node i, then print INF as the distance between i and j.

Example:
Input
```
4 4
1 2 1
2 3 1
3 4 1
4 1 1
```

Output:
```
0 1 2 1
1 0 1 2
2 1 0 1
1 2 1 0
```

Submission: Submit 1 C code <RollNo.>_fw.c