Operating Systems Lab Test

Q1. **Asynchronous Distributed Mean Computation(ADM)**

In this question you will compute the mean in a graph of nodes using an synchronous distributed algorithm. Let's say you have a connected undirected graph consisting of N nodes. Each node has some value. You need to find the mean of all these values and this mean should reach to all the nodes. The nodes communicate with each other till they are able to find the mean value. The algorithm followed is described below (label propagation).

Algorithm

The algorithm takes place in many iterations. Each node has a unique id. Each node maintains its own value and a list containing the values of other nodes. This list keeps growing with each iteration. Initially the list is empty for each node. In the first iteration, every node sends its own value to all its neighbours. Each node stores the values received from the neighbours in its own list.

Now, in every subsequent iteration, each node sends to all its neighbours, the values which were **newly added to it's own list in the last iteration**. Each node updates its own list based on the values it receives from its neighbours. After some number of iterations, the list of items (values) in all the nodes becomes same. If D is the diameter of the graph, then D number of iterations is sufficient. Let's make it easier for you. If N is the number of nodes in the graph, then **N number of iterations is also sufficient**.

In order to keep the computation simple, the iterations will be done in an asynchronized way, i.e. if a node has completed its i^{th} iteration, then it will start a next iteration immediately, even though some other node is still in its i^{th} iteration. No need to synchronize.

Implementation
You need to implement this computation using **signals and pipes only**.
Let the input graph have N nodes. There will be a master process (**master.c**). The master process creates N child/node processes (exec **node.c**) and sends them random values between 1 - 100. It also creates 2 pipes corresponding to each edge in the graph. If e(u,v) is an edge, then one pipe is used for communication from process u to v and other pipe is used to communicate from process v to u. Therefore child processes represent the nodes and pipes represent the edges.

The processes communicate with each other according to the algorithm described above. After N iterations, all the processes compute the mean using their own value and their own list. Next, they exit and return the computed mean as the exit status. The master process needs to assert that it has received the same mean value from all the N processes and that it is the correct mean in the graph.

Input Format: First line contains 2 integers N and M. N is the number of nodes. M is the number of undirected edges. Then M lines follow each containing 2 space separated integers. Line i+1 contains 2 integers u_i and v_i (1-indexed) which represents an undirected edge between nodes u_i and v_i.

Output Format: The first line prints the initial values of all the nodes. The second line prints the mean values received at the end from all the nodes. Printing is done by **master process**.

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

Output
Initial values of nodes:(Pid1, 20); (Pid2,10); (Pid3, 5); (Pid4, 30)
Final mean values received from nodes: (Pid1,16) (Pid2,16) (Pid3,16) (Pid4,16)

Where Pid1, Pid2, Pid3 and Pid4 represent the pids of the 4 processes.

Submission: Submit 2 C codes. <RollNo.>_master.c which represents the master process. <RollNo.>_node.c which represents the node process.

Bonus: Instead of using N iterations, use only D (diameter) iterations and solve the problem.