BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
Work Integrated Learning Programmes Division
SECOND SEMESTER 2014-15
Distributed Computing (SS ZG 526)

EC-1 (1st Component: 8%) **Date of Submission: 04/03/2015**

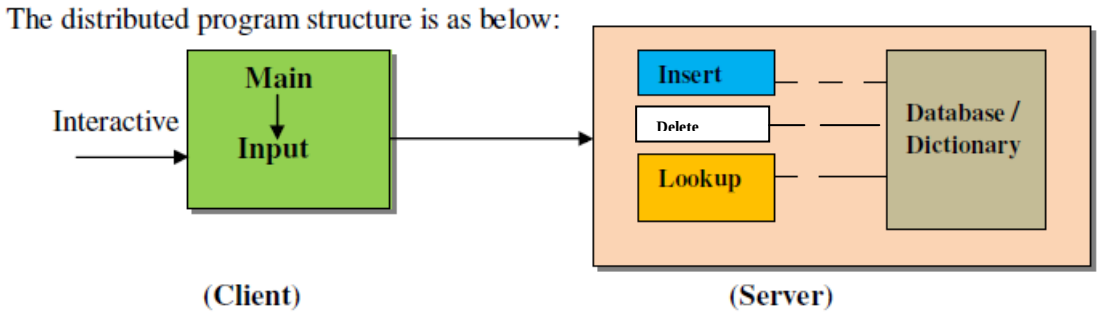
Q.1 The goal of this problem is to give you programming exposure to Remote Procedure Calls, abbreviated as RPCs, which is an implicit way of communication in distributed systems. In building a distributed application, RPC systems make the task of network programming much easier (without worrying about Socket APIs or send/ receive primitives).

Your job is to write a distributed application for Dictionary Lookup using **rpcgen** utility covered in one of the online sessions. You should write an RPC client, and a RPC server.

The services that should be provided by the remote procedure are:

1) Inserting words in to a dictionary along with its meaning (multiple words).
2) Lookup to search for a word in the dictionary. If found return the meaning of the word.
3) Delete a word alongwith its' meaning from the dictionary.

The input to the client program should be interactive; for example, each request may contain the Procedure name, and the Word. Say if you call insert with a word, server should return success or if you call lookup with a word, server should return its associated meaning. Input is to be taken from the keyboard till the client wants to exit. The dictionary should not contain multiple entries for the same word. A word should have only one meaning. Devise a suitable data structure for the dictionary. You are free to make any other assumptions if you feel these are necessary.



Q.2 In this question, you need to implement Lamport's Logical Clocks (scalar time) for a distributed computing system. Construct the implementation rules for updating clock for local events and also for message send and receive events. You may use the following as guideline:

- Each process keeps an integer, initially 0 that represents its internal logical clock.
- Whenever a process takes a local step, it increments its logical time by 1, and the incremented time is considered to be the time of the local event.
- Whenever a process sends a message (send event), it increments its logical time by 1, and sends that new time with the message. This time is considered to be the logical time of the send event.
- Whenever a process receives a message, it first compares its own logical clock time to the logical time sent with the message, and sets its own logical clock to be the maximum of the two times. Then, it increments its logical time by 1, and the incremented time is considered to be the time of the receive event.

Test your implementation by having the modules take internal steps and send messages randomly. Each time an event occurs at a module, have it print out the logical time of the event. Check that the logical times assigned to your events respect the happened before relationship. Each module can be implemented as a thread to represent one independent node in the distributed computing platform. Make any other assumption if you feel it is needed. You can chose any language of your choice for implementing 2nd question, but the first one should be in Sun RPC (over any Unix or Linux) platform.
