1. (a) Write a server function that receives a message and sends it back to the sender. Also write a client function that sends a message, waits for the reply, and outputs the returned message. Start two processes running these functions.

(b) Modify the server function so that it does not simply echo the received message, but operates as follows: if the message is a number \( N \), it returns the \( N \)th fibonacci number; if the message is not a number, it returns an error message to the client.

2. (a) Write a function master for a process that understands and reacts to the following messages:
   - **create**: spawns a new slave process.
   - \{send,\( X \)\}: sends message \( X \) to all slave processes.
   - **kill**: terminates all slave processes.
   - **terminate**: terminates the master process after having terminated all the slave ones.

In addition, write a function slave that understands and reacts to the following messages:
   - **terminate**: terminates the process
   - \( X \): prints on the screen message \( X \), together with the process id of the slave itself.

The master process needs to keep information about all the slave processes it created so far.

(b) Modify the above functions so that the master process monitors the slave processes, i.e., it links to each slave process and traps potential exit signals. If the master process receives an exit signal, it removes the slave process from the list of active slave processes, and respawns a new slave process. The reaction to the message **kill** also has to be modified: before terminating the slave processes, they must be unlinked.

*Hint:* check out functions delete/2 and unlink/1.