```
/*
 Master pseudo code

 General Data-Structures

 Map bankServerChain { bankId, List<Servers> }
 Map bankServer { bankId, head, tail }
 Map bankClients { bankId, clientId }
 List serverTS { serverId, timestamp }
 enum serverType { Head, Internal, Tail }
 enum serverRelation { successor, predecessor }

 Events:
        - receive
                - The server sends the liveness update.
                - The master updates the serverTS list with its own current TS for the corresp
onding server.
                - We assume that the master and server clocks are in sync (const drift).

        - extendChain
                - The new server wants to be added in the chain for a bank.

 Functions:
        - notifyAll
                - notify the client of the new head/tail, in case of head/tail failure.
        - notifyServer
                - notify the server of the new successor/predecessor for internal failure.
                - also notify the respective servers if they are new head/tail, in case of hea
d/tail failure.

        - checkFailure
                - probe the list of servers (every second), calculate the difference between t
he currentTS and the TS
                - of each server. If the difference is greater than 5 sec then its a server fa
ilure as no liveness
                - update was received by the master for that particular server in last 5 secs.

        - handleFailure
                - Handles failure for head, tail and internal servers

*/
        /* Load the constants from the config file */

        // callback to receive the heart beat notification from the server
        // and update the serverTS list
        event receive(serverId):
                synchronize(serverTS) {         // take lock on the serverTS list
                        updateTS(serverId, currTS);
                }
        end

        // callback function to receive the extend chain notification
        // from a new server
        event extendChain(serverId, bankId):
                oldTail = updateNewTail(bankId, serverId);      // update the tail info in the
 data structure
                notifyHeadTailServer(serverId, serverType:Tail);        // notify the new serv
er that it is tail
                notifyInternalServer(oldTail, serverId, serverRelation:successor); // notify t
he old tail of its successor
                notifyAll(bankId, serverId, serverType:Tail);   // notify the client of the ne
w tail
        end
```

```
        // function to notify the client regarding the new head/tail server
        function notifyAll(bankId, serverId, type [head/tail]):
                broadCast(bankId, serverId, type);  // Notify all the clients/servers correspo
nding to the bankId.
        end

        // function to notify the successor/predecessor server of the failure
        // of the internal server
        function notifyInternalServer(serverId):
                succ = findSuccessor(serverId);
                pred = findPredecessor(serverId);
                seqNum = sendServer(succ, pred, serverRelation:predecessor); // get the last s
eqNum received from the successor
                sendServer(pred, succ, serverRelation:successor, seqNum); // send the sequence
 number to the predecessor
        end

        // function to notify the server that they are new head/tail
        function notifyHeadTailServer(serverId, type [head/tail]):
                sendServer(serverId, type);
        end

        // periodically check whether there is any failed server
        // depending upon the received time stamp
        function checkFailure():                // Will be called every sec
                synchronize(serverTS) {         // lock the serverTS list and probe
                for server in serverTS:
                        if(currentTS-server.serverTS > 5):      // Implies that the master did
 not receive any notification
                                                                // from server in last 5 sec
                                type = findServerType(serverId);
                                handleFailure(serverId, type);
                }
        end

        // fucntion to handle the failure of servers head/tail/internal
        function handleFailure(serverId, type):
                switch(type):
                case Head:
                        head = updateNewHead(serverId); // update the head in the data structu
re, will return the new head
                        notifyAll(bankId, head, serverType:Head);       // notify the client o
f the new head
                        notifyHeadTailServer(head, serverType:Head);    // notify the new head
 that it is head
                        break;
                case Tail:
                        tail = updateNewTail(serverId); // update the tail in the data structu
re, will return the new tail
                        notifyAll(bankId, tail, serverType:Tail);       // notify the client o
f the new tail
                        notifyHeadTailServer(tail, serverType:Tail);    // notify the new tail
 that it is tail
                        break;
                case Internal:
                        updateServerList(serverId);     // Remove the internal server from the
 list
                        notifyInternalServer(serverId); // Inform the internal server about it
s new sucessor and predecessor
                        break;
        end
```