MODULE *RaftHeartbeat*

EXTENDS $\mathit{Naturals}, \mathit{FiniteSets}, \mathit{Sequences}, \mathit{TLC}$

Is leader *ALIVE* or *CRASHED*

VARIABLE $\mathit{leaderState}$

A collection of heartbeat (*AppendEntries*) messages the leader has sent.
A single message is abstracted to represent the leader's index

VARIABLE $\mathit{messages}$

A representation of the *commitIndex* and term, leader increases index monotonically.

VARIABLE $\mathit{leaderIndex}$
VARIABLE $\mathit{followerIndex}$
$\mathit{nodeIndexes} \triangleq \langle \mathit{leaderIndex}, \mathit{followerIndex} \rangle$

Indicates whether the follower timed out after not hearing from
the leader for the specified amount of time.

VARIABLE $\mathit{isTimeout}$

$\mathit{vars} \triangleq \langle \mathit{leaderState}, \mathit{messages}, \mathit{nodeIndexes}, \mathit{isTimeout} \rangle$

Optional values to limit the state-space complexity of the model

CONSTANTS $\mathit{MESSAGE\_LIMIT}, \mathit{LEADER\_INDEX\_LIMIT}$

Check whether the message limit is reached

$\mathit{IsOverMessageLimit} \triangleq \mathit{Len}(\mathit{messages}) \geq \mathit{MESSAGE\_LIMIT}$

Check whether the index limit is reached

$\mathit{IsOverLeaderIndexLimit} \triangleq \mathit{leaderIndex} \geq \mathit{LEADER\_INDEX\_LIMIT}$

The leader crashes and doesn't recover

```
CrashLeader ≜
        $\land\ \mathit{leaderState} = \text{“ALIVE”}$
        $\land\ \mathit{leaderState}' = \text{“CRASHED”}$
        $\land$ UNCHANGED $\langle \mathit{messages}, \mathit{nodeIndexes}, \mathit{isTimeout} \rangle$
```

The leader sends the follower an *AppendEntries* message

```
SendMessage ≜
        $\land\ \mathit{leaderState} = \text{“ALIVE”}$
        $\land\ \mathit{messages}' = \mathit{Append}(\mathit{messages}, \mathit{leaderIndex})$
        $\land$ UNCHANGED $\langle \mathit{leaderState}, \mathit{nodeIndexes}, \mathit{isTimeout} \rangle$
```

Helper function to remove a message from a sequence of messages

```
RemoveMessage(i, seq) ≜
    $[j \in 1 \,..\, \mathit{Len}(\mathit{seq}) - 1 \mapsto$ IF $j < i$ THEN $\mathit{seq}[j]$ ELSE $\mathit{seq}[j+1]]$
```

The network drops a message

$\mathit{DropMessage} \triangleq$

```
        ∧ $Len(messages) \geq 1$
        ∧ $\exists\, i \in 1\,..\,Len(messages)$ :
            $messages' = RemoveMessage(i,\ messages)$
        ∧ UNCHANGED $\langle leaderState,\ nodeIndexes,\ isTimeout \rangle$
```

The leader increments its index

```
IncrementIndex ≜
        ∧ $leaderState = $ "ALIVE"
        ∧ $leaderIndex' = leaderIndex + 1$
        ∧ UNCHANGED $\langle leaderState,\ messages,\ followerIndex,\ isTimeout \rangle$
```

The follower receives a message from the leader.

```
ReceiveMessage ≜
        ∧ $Len(messages) \geq 1$
        ∧ $\exists\, i \in 1\,..\,Len(messages)$ :
            ((LET $message \triangleq messages[i]$
            IN    $followerIndex' = $ IF $message > followerIndex$
                                     THEN $message$
                                     ELSE $followerIndex$)
             ∧    $messages' = RemoveMessage(i,\ messages)$)
        ∧ UNCHANGED $\langle leaderState,\ leaderIndex,\ isTimeout \rangle$
```

The follower times out

```
Timeout ≜ $isTimeout' = $ TRUE
         ∧   UNCHANGED $\langle leaderState,\ messages,\ nodeIndexes \rangle$
```

Initial state of model

```
Init ≜ ∧ $leaderState = $ "ALIVE"
       ∧ $messages = \langle\rangle$
       ∧ $leaderIndex = 0$
       ∧ $followerIndex = 0$
       ∧ $isTimeout = $ FALSE
```

Next state function

```
Next ≜ ∧ $\neg isTimeout$
       ∧  ∨ $(\neg IsOverMessageLimit \land SendMessage)$
          ∨ $(\neg IsOverLeaderIndexLimit \land IncrementIndex)$
          ∨ $DropMessage$
          ∨ $ReceiveMessage$
          ∨ $CrashLeader$
```

$$Spec \triangleq Init \land \Box[Next]_{vars}$$