Coordination in Distributed Systems

We want to realize election algorithms in Erlang.

We use process IDs (PIDs) as election values, which can be compared with $<$. (If we wanted to handle process substitution (on crash) self-assigned election values might be more appropriate.)

The required list operators (`max`, `append`, `member`) can be found in the `lists` module.
A process is considered to be unreachable, if is does not answer within a quarter of a second (250ms). Since we expect answers, simple asynchronous one way Erlang communication is not sufficient. Instead we can use synchronous communication e. g. via the function `rpc`.

```
rpc(Pid, Request) ->
    Pid ! {self(), Request},
    receive
       {Pid, Response} ->
           Response
       after 250 ->
           unreachable
    end.
```

1. **Bully Algorithm**
   Please write an Erlang module `bully`, that defines the behavior of processes, which use the Bully algorithm to elect a coordinator from a group of processes.

   The state of each process contains at least the current coordinator (maybe the own election value) and also a list of all other processes in the group.

   a) Please realize the basic functionality of the bully algorithm:
      - Sending of `election` messages to all process with a higher number,
      - Processing of `election` messages,
      - Sending and processing of `coordinator` messages (including the output of election results).

      By means of a `startElection` message, a process should be triggered to start a new election.

   b) Please write a function `setup`, which will construct a new group of processes.

   c) Test your coordinator election by sending `startElection` messages to selected processes. Terminate some processes and repeat the election. Start new processes and add them to the group.

   d) Please measure, how many `election` messages are sent, by sending a copy of each message to a counting process.

Please be prepared for face-to-face technical discussions. These will happen during the on-site tutorial sessions.