RPC Based Election Service

Specify an RPC interface to an Election Service which allows a client to both query the current number of votes for a specified candidates and vote for one of the set of candidates. Each client has a voter number used for identification in requests and candidates are identified by a string name.

Give a *pseudocode* implementation for the Election Service (server only) which would permit the interface to be invoked using an RPC mechanism. The RPC implementation supports *at-least-once* calling semantics but clients must only vote once. Explain the implications of the RPC semantics on your implementation of the election server.

Election Service

```
interface election {

        void vote  (in int voterid, in char* candidate);
        void query (in char* candidate, out int votecount);  }


include  election.idl

void main () {

    status = export ( election, "electserver", docnameserver);
    status = RPCServerListen ();

    voted:    array of booleans indexed by voterid of clients indicating
              whether they have voted  (could be a list);

    votes:    array of votes 'indexed' by candidate name

vote ( voterid, candidate) {
    if not (voted [voterid]) {
        voted [voterid] := true;
        votes [candidate]++
        } else do nothing as candidate has voted     }

query (candidates, votecount) {
    votecount := votes [candidate]; }
```

Both of these operations are idempotent in that one or more execution have the same effect, so they can be repeated. Consequently they can be used with at-least-once semantics