```
qdata Coin      = {Heads | Tails}
qdata Progress  = {Done | NotYet}
qdata List a    = {Cons (a, List( a)) | Nil}

toss ::( ; c:Coin) =
{  q = |0>;
   Had q;
   measure q of
     |0> => {c = Heads}
     |1> => {c = Tails}
}

checkForDone::(size:Int,possibleLdr:Int | lis: List(Int); done:Progress, res:List(Int),possibl
eLeader:Int)={
  case lis of
    Cons(hd,tl) => {
      use hd in {
        if (size == hd) => {
            done = Done;
            possibleLeader = possibleLdr;
            h = hd;
            rtl = tl;
         }
        else => {
          checkForDone (size, possibleLdr-1 | tl; done, rtl,possibleLeader);
          h = hd
        }
     };
     res = Cons(h,rtl);
     }
     Nil => {
       done = NotYet;
       res = Nil;
       possibleLeader = 0;
    }
}


elect::(size:Int|lis:List(Int); leader : Int) ={
  vote lis;
  checkForDone(size,size|lis; done, lis, possibleLdr);
  case done of
   Done => {
     leader = possibleLdr;
     discard lis}
   NotYet => {
     discard possibleLdr;
     leader = elect(size|lis)}
}

vote::(lis:List(Int); vtlis:List(Int)) ={
  case lis of
    Cons (hd, tl) => {
      case toss() of
         Heads => {
            subvote(hd,tl ; reslist, vt);
            vtlis = Cons(vt,reslist)
         }
         Tails => {
            subvote(0,tl ; reslist, vt);
            use hd,vt in { nh = hd + vt};
```

```
            vtlis = Cons(nh,reslist)
         }
    }
    Nil => {vtlis = Nil}
}

subvote::(last:Int, lis: List(Int); svlis:List(Int), vt:Int) = {
case lis of
  Cons(hd,tl) => {
    case toss() of
         Heads => {
            subvote(last,tl ; reslist, subvt);
            svlis = Cons(subvt,reslist);
            vt = hd;
         }
         Tails => {
            subvote(last,tl ; reslist, subvt);
            use hd,subvt in { nh = hd + subvt};
            svlis = Cons(nh,reslist);
            vt = 0
         }
    }
  Nil => {vt = last; svlis = Nil}

}


makeListOfOnes::(i:Int| ; mlis: List(Int)) = {
  if i == 0 => { mlis = Nil}
  else => {
     tl = makeListOfOnes(i-1|);
     mlis = Cons(1,tl)
  }

}
main::() =
{ lis = makeListOfOnes(5|);
  ldr = elect(5|lis)
}
```