```
function [V] = eg_7_5_learn_at(lambda,alpha,n_rw_states,nEpisodes)
% EG_7_5_LEARN_AT - Performs td(lambda) learning with {\em accumulating} eligibility
traces
%
% Written by:
% --
% John L. Weatherwax                2007-12-07
%
% email: wax@alum.mit.edu
%
% Please send comments and especially bug reports to the
% above email address.
%
%-----

% the number of episodes (in general this would need to be much larger to gaureentee
complete learning):
%nEpisodes = 1000;

% the number of non terminal states plus the one terminal states:
nStates=n_rw_states+1;

% initialize the states ... terminal states are defined to be ZERO
V = zeros(1,nStates);

% initialize the eligbility trace (to zero):
et = zeros(1,nStates);

%gamma     = 1.0;
gamma     = 0.9;

for ei=1:nEpisodes,

  % perform a random walk getting the sequence of rewards recieved and the
  % sequence of states we transition through ... this is the surragate of
  % performing actions with some policy
  %
  [rewseen,stateseen] = eg_7_5_episode(n_rw_states);

  % compute the temporal difference and elibility traces update for each
  % state s_t for t=0,1,2,\cdots,T-1 (the states visited)
  %
  % and use these to update V(\cdot):
  %
  T = length(rewseen);
  for t=0:(T-1),    % <- STATE index ...

    st     = stateseen(t+1);
    rew    = rewseen(t+1);
    stp    = stateseen(t+2);
    delta  = rew + gamma*V(stp) - V(st); % the temporal difference
    % implement UPDATING elagability traces:
    et(st) = et(st)+1;

    % update V(\cdot) with this information:
    V = V + alpha*delta*et;
    % decay our elibility trace:
    et = gamma*lambda*et;
```

```
  end % end state sequence loop ...

end % end episode loop ...

% remove the terminal states:
V = V(1:end-1);
```