```
function [Q] = rr_action_bellman(alpha,beta,gamma,Rsearch,Rwait)
% RR_ACTION_BELLMAN - Seeks an iterative solution to the optimal Bellman equations for
the action-value function.
%
% Inputs:
%
% State Transition Probabilities:
%
% alpha = the probability that after searching with a "high" energy level we end with a
"high"
% beta  = the probability that after searching with a "low" energy level we end with a
"low"
% gamma = the learning rate
%
% Recieved Rewards (Rsearch > Rwait):
%
% Rsearch = the reward given to searching
% Rwait   = the reward given to waiting    (Rwait <= Rsearch)
%
% See ePage 226-227 for the Bellman equation for the optimal state-value function Q^*(s)
and
% the book solutions for the Bellman equation for the optimal action-value function
Q^*(s,a).
%
% Written by:
% --
% John L. Weatherwax                2007-12-03
%
% email: wax@alum.mit.edu
%
% Please send comments and especially bug reports to the
% above email address.
%
%-----

if( nargin==0 )
  % State Transition Probabilities:
  %
  % the probability that after searching with a "high" energy level we end with a "high"
  alpha   = 0.8;
  % the probability that after searching with a "low" energy level we end with a "low"
  beta    = 0.2;
  % the learning rate:
  gamma   = 0.9;

  % Recieved Rewards (Rsearch > Rwait):
  %
  Rsearch = 2.0; % <- the reward given to searching
  Rwait   = 1.0; % <- the reward given to waiting
end

% Initialize our action-value function:
% state\action:      search       wait      recharge
%  high                                       N.A.
%  low
Q = zeros(2,3);
% some convience variables denoting states and actions:
h = 1; l = 2;
s = 1; w = 2; rc = 3;
```

```
% some parameters for convergence:
%
MAX_N_ITERS = 100; iterCnt = 0;
CONV_TOL    = 1e-3; delta = 1e10;

while( (delta > CONV_TOL) && (iterCnt <= MAX_N_ITERS) )
  delta = 0;
  % update action states in the following order:
  %
  q = Q(h,s);
  Q(h,s) = Rsearch + alpha*gamma*max( [ Q(h,s), Q(h,w) ] ) + (1-alpha)*gamma*max( [
Q(l,s), Q(l,w), Q(l,rc) ] );
  delta = max( [ delta, abs( q-Q(h,s) ) ] );

  q = Q(h,w);
  Q(h,w) = Rwait + gamma*max( [ Q(h,s), Q(h,w) ] );
  delta = max( [ delta, abs( q-Q(h,w) ) ] );

  q = Q(l,s);
  Q(l,s) = (1-beta)*(-3 + gamma*max( [ Q(h,s), Q(h,w) ] ) ) + beta*( Rsearch + gamma*max(
[ Q(l,s), Q(l,w), Q(l,rc) ] ) );
  delta = max( [ delta, abs( q-Q(l,s) ) ] );

  q = Q(l,w);
  Q(l,w) = Rwait + gamma*max( [ Q(l,s), Q(l,w), Q(l,rc) ] );
  delta = max( [ delta, abs( q-Q(l,w) ) ] );

  q = Q(l,rc);
  Q(l,rc) = gamma*max( [ Q(h,s), Q(h,w) ] );
  delta = max( [ delta, abs( q-Q(l,rc) ) ] );

  %Q, delta
  iterCnt=iterCnt+1;
end
```