Computational Approach to solving Hopenhayn-Rogerson:

1. Initialize the algorithm: Set parameters, grid bounds, number of grid points. Only state variable is s, firm's productivity shock. Initialize exit policy function ($x(s)$) and value function ($W(s)$). Conjecture a price that clears entry market and a mass of entrants.

2. Solve for price that clears entry market:

   a. *Solve VFI*: Very standard. Only (dynamic) choice the firm makes is to exit or not.

   b. *Solve Entrant's Value*: $\int W(s;p)d\nu(s)$

   – If $|\int W(s;p)d\nu(s) - pc_e| < \varepsilon_p$, you are done. Otherwise, update good price and solve 2a - 2b again.

3. Solve for mass that clears Labor Market Clearing condition:

   a. *Solve for Stationary Distribution*: Note, this is nearly identical to how you calculated the stationary distribution in PS2

   b. *Solve for Labor Demand and Labor Supply*

   – If $|L_d - L_s| < \varepsilon_m$, you are done. Otherwise, update entrant mass and solve 3a - 3b again.

   – Note: An alternative way to solve (3) is to exploit the linear homogeneity of distribution equation to solve in "one shot". Computationally, this doesn't matter here, however for larger (harder) problems, it may be computationally more efficient to exploit the linear homogeneity.

A helpful hint for solving the model with random disturbances:

- Utility is given by

$$U_1(s;p) = \frac{\gamma_E}{\alpha} + \frac{1}{\alpha} ln\left(\sum_{x'} exp\left\{\alpha V^{(x')}(s;p))\right\}\right)$$

  which your computer may return as infinity under certain parameterizations.

- Use the Log-Sum-Exp trick:

$$\begin{aligned} y &= ln\left(\sum_n exp\{x_n\}\right) \\ &= c + ln\left(\sum_n exp\{x_n - c\}\right) \end{aligned}$$

Algorithm 1 Pseudo Code to Solve Hopenhayn-Rogerson

procedure HOPENHAYN-ROGERSON
    $p_0 = p^{init}$
    convergence flag price $= 0$
    while convergence flag price $= 0$ **do**
        call VFI( ) **return** $\{W(s;p_0), x(s;p_0)\}$
        call ENTVAL( ) **return** $\int W(s;p_0)d\nu(s)$

        if $|\int W(s;p_0)d\nu(s) - p_0c_e| > \varepsilon_p$ **then**   ▷ Hint: Update price similar to PS2
            if $\int W(s;p_0)d\nu(s) > p_0c_e$ **then**
                $p_1 < p_0$
            else if $|\int W(s;p_0)d\nu(s) - p_0c_e| < \varepsilon_p$ **then**
                $p_1 > p_0$
            end if
            $p_0 \leftarrow p_1$
        else if $|\int W(s;p_0)d\nu(s) - p_0c_e| < \varepsilon_p$ **then**
            convergence flag price $= 1$
        end if
    end while

    $m_0 = m^{init}$
    convergence flag mass $= 0$
    while convergence flag mass $= 0$ **do**
        call STATDIST( ) **return** $\mu(s;m)$
        call LMC( ) **return** $\{L_d(\mu,m;p), L_s(\mu,m;p)\}$

        if $|L_d(\mu,m;p) - L_s(\mu,m;p)| > \varepsilon_m$ **then**   ▷ Hint: Update mass similar to PS2
            if $L_d > L_s$ **then**
                $m_1 < m_0$
            else if $L_d < L_s$ **then**
                $m_1 > m_0$
            end if
            $m_0 \leftarrow m_1$
        else if $|L_d(\mu,m;p) - L_s(\mu,m;p)| < \varepsilon_m$ **then**
            convergence flag mass $= 1$
        end if
    end while
end procedure

function VFI( )
  for $i_s = 1 : n_s$ **do**
    Get $n^*(s)$ (solve firm's static labor problem)

    $W^{(x=0)}(s;p) = \pi(s;p) + \beta \int W(s';p)dF(s)$
    $W^{(x=1)}(s;p) = \pi(s;p)$
    $W(s;p) = \max\{W^{(x=0)}(s;p), W^{(x=1)}(s;p)\}$
  end for
  return $\{W(s;p_0), x(s;p_0)\}$
end function

function EntVal( )
  return $\int W(s;p)d\nu(s)$
end function

function StatDist( )
  $\mu'(s;p,m) = \int_S [1-x(s;p)]F(s'|s)d\mu(s;m) + m\int_S [1-x(s;p)]F(s'|s)d\nu(s)$

  Iterate until $\mu'(s;p,m) = \mu(s;m)$
  return $\mu(s;m)$
end function

function LMC( )
  $L_d = \int_S n^*(s;p)d\mu(s;m) + m\int_S n^*(s;p)d\nu(s)$
  $\Pi = \int_S \pi(s;p)d\mu(s;m) + m\int_S \pi(s;p)d\nu(s)$
  $L_s = 1/A - \Pi$
  return $\{L_d(\mu,m;p), L_s(\mu,m;p)\}$
end function

Algorithm 2 Pseudo Code to Solve Hopenhayn-Rogerson (Linear Homogeneity)

procedure Hopenhayn-Rogerson
  Solve for price that clears entry market exactly as in Algorithm 1.

  call StatDist( ) **return** $\mu(s;m=1)$
  call LMC( ) **return** $\{L_d(\mu,m=1;p), L_s(\mu,m=1;p)\}$

  $L_d(\mu,m^*;p) = m^* L_d(\mu,m=1;p)$ ▷ Linear Homogeneity
  $L_s(\mu,m^*;p) = 1/A + m^*\Pi(\mu,m=1;p)$ ▷ Linear Homogeneity

  $m^* = (A[L_d(\mu,m=1;p) + \Pi(\mu,m=1;p)])^{-1}$

  call StatDist( ) **return** $\mu(s;m=m^*)$
end procedure