Computational Approach to solving Transition Paths (Consea-Krueger Model):

1. Initialize the algorithm: Set parameters, grid bounds, number of grid points. Guess a length of time for the transition. The states of the economy are transition time t, age j, assets a and productivity z. Read in policy/value functions, stationary distribution, and aggregate allocations/prices from steady state with $\theta > 0$ and $\theta = 0$. Initialize asset and labor policy function and value function, distribution (μ_t) , and a transition path for the aggregate variables (K and L). Given the aggregate variables, we can derive the prices (w, b and r).

   - Note: Distribution at $t = 1$ will be same as Distribution when $\theta > 0$ — ie $\mu_1 = \mu_{\theta>0}$ — and Policy/Value function will be same as Policy/Value function when $\theta = 0$ — ie $V_T = V_{\theta=0}, a_T = a_{\theta=0}, l_T = l_{\theta=0}$. These won't change when you run your algorithm.

2. Shoot Backward:

   - Backward Induction: Take as given path for aggregate variables (K and L) – and thus by extension prices (w, b and r). Solve household's decision problem taking as given next age's and next time value function $V_{t+1}(j+1, a', z')$. Once done, you should get out value $V_t(j, a, z)$, asset policy function $a_t(j, a, z)$, and labor policy function $l_t(j, a, z)$ at each stage of the agent's life (j) and at each point in time along the transition path t.

3. Shoot Forward

   - Transition Distribution: Take as given policy function solved for in (2). Calculate the asset distribution following the law of motion.

$$\mu_{t+1}(j+1, a', z') = \int_{A,Z} 1_{a'_t = g^a_t(j,a,z)} dF(z'|z) d\mu_t(j, a, z)$$

   - Capital and Labor Market Clearing: Taking as given the policy function solved for in (2) and distribution solved for in (3), calculate new aggregate capital and labor

$$K_{t+1} = \sum_J \int_{A,Z} a d\mu_{t+1}(a, z)$$
$$L_{t+1} = \sum_J \int_{A,Z} g^l_{t+1}(j, a, z) d\mu_{t+1}(a, z)$$

If transition path has converged – that is, if "new" Capital and Labor path (approximately) equal "guess" Capital and Labor path – you are done. Otherwise, 'update' capital and labor path in the same way you did last week. Then repeat steps (2) - (3). Once transition path has converged, check to make sure it has converged to the new steady state: If $|K_T - K_{\theta=0}| + |L_T - L_{\theta=0}| < \delta$, you are done. If not, increase T and start again.

Pseudo Code to Solve Transition Paths (Consea-Krueger Model):

Algorithm 1 Transition Paths (Consea-Krueger)

procedure MAIN CODE
    call READCONESAKREUGER($\theta > 0$)
    call READCONESAKREUGER($\theta = 0$)

    $K_t^0 = [K_{\theta>0}, K_{\theta>0} + \Delta, \ldots, K_{\theta=0}]$ ▷ Δ is a linear step
    $L_t^0 = [L_{\theta>0}, L_{\theta>0} + \Delta, \ldots, L_{\theta=0}]$ ▷ Δ is a linear step
    Given $\{K_t^0, L_t^0\}$ solve for $\{w_t^0, r_t^0, b_t^0\}$

    convergence flag $= 0$
    while convergence flag $= 0$ **do**
        call SHOOTBACKWARD( )
        return $\{V_t(j,a,z), a_t'(j,a,z), l_t(j,a,z)\}$

        call SHOOTFORWARD( )
        return $\{\mu_t(j,a,z), K_t^1, L_t^1\}$

        if $|K_t^1 - K_t^0| + |L_t^1 - L_t^0| > \epsilon$ **then** ▷ Try $\lambda = 0.5$
            $K_t^0 \leftarrow \lambda K_t^1 + (1-\lambda)K_0$
            $L_t^0 \leftarrow \lambda L_t^1 + (1-\lambda)L_0$

        else if $|K_t^1 - K_t^0| + |L_t^1 - L_t^0| < \epsilon$ **then**
            if $|K_T^1 - K_{\theta=0}| + |L_T^1 - L_{\theta=0}^0| < \delta$ **then**
                return convergence flag $= 1$
            else if $|K_T^1 - K_{\theta=0}| + |L_T^1 - L_{\theta=0}^0| > \delta$ **then**
                Increase T and start algorithm over again.
            end if
        end if

    end while
end procedure

```
function ShootBackward( )
    for t = T − 1 : −1 : 1 do
        call BackwardInduction( )
    end for
    return {V_t(j, a, z), a'_t(j, a, z), l_t(j, a, z)}
end function

function BackwardInduction( )
    for j = j_N : −1 : 1 do                                              ▷ Backward Induction
        for a = 1 : n_a;   z = 1 : n_z do
            Solve HH Problem       ▷ Separate problem for end of life, retirees, and workers
            V_t(j, a, z) = u(c, l) + βE_{z'}{V_{t+1}(j + 1, a', z')}      ▷ Continuation Value
        end for
    end for
    return {V_t(j, a, z), a'_t(j, a, z), l_t(j, a, z)}
end function


function ShootForward( )
    for t = 1 : T − 1 do
        call CalculateDist( )
        call UpdatePath( )
    end for
    return {μ_t(j, a, z), K^1_t, L^1_t}
end function

function CalculateDist( )
    Construct Π_t^{n_a n_z × n_a n_z}(j)                  ▷ Π depends on agent's age and time
    Update μ_{t+1}(j + 1, a, z) = Π_t(j)' μ_t(j, a, z) 1/(1+n)
    return μ_{t+1}(j, a, z)
end function

function UpdatePath( )
    Aggregate K^1_{t+1} = Σ_J ∫_{A,Z} a dμ_{t+1}(a, z)
    Aggregate L^1_{t+1} = Σ_J ∫_{A,Z} g^l_{t+1}(j, a, z) dμ_{t+1}(a, z)
    return {K^1_{t+1}, L^1_{t+1}}
end function
```