```
function out=CalculaRendaGov(tau_y, Labor, WageShocks, eco_param, report)

    % Fill in unset optional values.
    switch nargin
        case {4}
            report = 0;
    end

    eps = 1e-5;

    eco_param.tau_y = tau_y;

    fprintf('tau_y:%1.3f\ttau_n:%1.3f\n',eco_param.tau_y, eco_param.tau_n)

    for nContador =1:100
        %% Definir um valor para $r_j \in (-\delta, 1/\beta -1)$;
        eco_param.r = (eco_param.r_UpperBound + eco_param.r_LowerBond)/2;

        %% Determinar o capital-labor ratio $r = F_K(k) -\delta$
        k = ((eco_param.r + eco_param.delta)/((1-eco_param.tau_y)*eco_param.alpha))^(1/↙
(eco_param.alpha -1));

        %% Determinar w:  $w = F_L(k)$;
        w = ((1-eco_param.tau_y)/(1+eco_param.tau_n) )*(1 - eco_param.alpha)*k^↙
(eco_param.alpha);

        %% Resolver o problema dos consumidores e determinar $a_{t+1}(a;z)$, $c_t(a;z)$,↙
$l_t(a;z)$;

        % Como temos um limite natural do ativo vamos definir os grids.
        Asset.Grid.N = Labor.Grid.N;
        Asset.Grid.Min = - w * WageShocks.Grid.Min/eco_param.r;
        Asset.Grid.Max = 100;
        Asset.Values = linspace(Asset.Grid.Min, Asset.Grid.Max, Asset.Grid.N);

        [~, ~, Policy] = SolveConsumerProblem(Asset, Labor, WageShocks, w, eco_param);

        %% Calcular a distribuição estacionária $\lambda(a;z)$
        Lambda = ConstructLambda(Policy, Asset, WageShocks);

        %% Calcula a oferta de capital agregado e oferta de trabalho
        K = Lambda(:)' * Policy.Asset.Values(:);
        L = Lambda(:)' * Policy.Wages.Values(:);
        Demanda = k - K/L;

        %%  Se $D(r) > 0$, então $r_{j+1} > r_j$; se $D(r) < 0$, então $r_{j+1} < r_j$.

        if abs(Demanda) < eps
            break;
        elseif Demanda < -eps
            eco_param.r_UpperBound = eco_param.r;
```

```
        elseif  Demanda > eps
            eco_param.r_LowerBond = eco_param.r;
        end

        if report == 1
            fprintf('Inter:%4d\tr: %1.6f\tDem: %2.6f\n', nContador, eco_param.r,↙
Demanda);
        end
    end
    Y = K^eco_param.alpha*L^(1-eco_param.alpha);
    out = eco_param.tau_y * Y + eco_param.tau_n*L*w;

    if report == 1

        fprintf('interest rate: %f\nwage rate: %f\n',eco_param.r, w);
        array2table(Policy.AssetPrime.Values)
        array2table(Policy.Labor.Values)
        array2table(Lambda)

        output = Policy.Asset.Values .^ eco_param.alpha + Policy.Labor.Values .^ (1-↙
eco_param.alpha);

        capital_output = Policy.Asset.Values ./ output;

        [sortedK, indexK] = sort(capital_output(:));
        lambda_aux = Lambda(:);

        figure
        plot(sortedK, cumsum(lambda_aux(indexK)));
        title('Capital-Output Ratio');
        xlabel('Capital-Output Ratio');
        ylabel('Cumulative sum');


        [sortedK, indexK] = sort(Policy.Wealth.Values(:));
        lambda_aux = Lambda(:);
        figure
        plot(sortedK, cumsum(lambda_aux(indexK)));
        title('Wealth');
        xlabel('Wealth');
        ylabel('Cumulative sum');
    end

end % end of function
```