```
function minima = gold_section(x0, step_size)

Ex = 0.0001;
% Convergence criteria

[xL, xR] = bounding_phase_algo(x0, step_size);
% defining x_left and x_right from the bounding phase algorithm

L_init = xR - xL;
L = [L_init];
% storing the initial value of L which is needed to check the convergence
% criteria

gamma = 0.618;

i = 1;

x1 = gamma*xL + (1-gamma)*xR;
x2 = (1-gamma)*xL + gamma*xR;
% values for the first iteration

while true

    x = [xL x1 x2 xR]
    % storing values in a matrix form for easier accessibility

    f1 = objF(x1);
    f2 = objF(x2);

    for j = 1:4
        f(j) = objF(x(j));
    end
    f

    % checking whether the function is increasing or decreasing by
    % comparing values of f1 and f2
    % Values of xL, x1, x2 and xR are updated and stored in each iteration

    if f1 > f2
        xL = x1;
        x1 = x2;
        xR = xR;
        x2 = (1-gamma)*xL + gamma*xR;
        x = [xL x1 x2 xR];
        x_opt = (xL + xR)/2
        f_opt = objF(x_opt)
    elseif f2 > f1
        xL = xL;
        xR = x2;
```

```
        x2 = x1;
        x1 = gamma*xL + (1-gamma)*xR;
        x = [xL x1 x2 xR];
        x_opt = (xL + xR)/2
        f_opt = objF(x_opt)
    end

    % adding new values of L to the matrix
    L = [L, xR - xL];

    % checking the convergence criteria
    if (L(i)/L_init) <= Ex
        minima = x_opt;
        break
    else
        i = i+1;
        continue
    end
end

end
```