```
function cec_optimization
    %% Configurations
    funcs = {@sphere_func, @rastrigin_func, @rosenbrock_func};
    func_names = {'Sphere', 'Rastrigin', 'Rosenbrock'};
    D_values = [2, 10];

    for d = 1:length(D_values)
        D = D_values(d);
        fprintf('\n========= DIMENSION: D = %d =========\n', D);

        for f = 1:length(funcs)
            func = funcs{f};
            fname = func_names{f};
            [lb, ub] = get_bounds(fname, D);
            fprintf('\n--- Function: %s ---\n', fname);

            fprintf('Running GA...\n');
            ga_stats = run_ga(func, D, lb, ub);
            print_stats('GA', ga_stats);

            fprintf('Running PSO...\n');
            pso_stats = run_pso(func, D, lb, ub);
            print_stats('PSO', pso_stats);

            fprintf('Running SA...\n');
            sa_stats = run_sa(func, D, lb, ub);
            print_stats('SA', sa_stats);
        end
    end
end

%% === Function Definitions ===
function f = sphere_func(x)
    f = sum(x.^2);
end

function f = rastrigin_func(x)
    f = 10 * numel(x) + sum(x.^2 - 10 * cos(2 * pi * x));
end

function f = rosenbrock_func(x)
    f = sum(100 * (x(2:end) - x(1:end-1).^2).^2 + (x(1:end-1) - 1).^2);
end

function [lb, ub] = get_bounds(fname, D)
    switch fname
        case 'Sphere',     lb = -100 * ones(1, D); ub = 100 * ones(1, D);
        case 'Rastrigin',  lb = -5.12 * ones(1, D); ub = 5.12 * ones(1, D);
        case 'Rosenbrock', lb = -30 * ones(1, D); ub = 30 * ones(1, D);
    end
end

function print_stats(name, stats)
```

```
    fprintf('%s - Best: %.4e | Worst: %.4e | Mean: %.4e | Std: %.4e\n', ...
        name, stats(1), stats(2), stats(3), stats(4));
end

%% === Optimizers ===

function stats = run_ga(func_handle, D, lb, ub)
    results = zeros(15,1);
    for i = 1:15
        rng(i);
        options = optimoptions('ga', 'MaxGenerations', 100, 'PopulationSize', 50,
'Display', 'off');
        [~, fval] = ga(func_handle, D, [], [], [], [], lb, ub, [], options);
        results(i) = fval;
    end
    stats = [min(results), max(results), mean(results), std(results)];
end

function stats = run_pso(func_handle, D, lb, ub)
    results = zeros(15,1);
    for i = 1:15
        rng(i);
        options = optimoptions('particleswarm', 'SwarmSize', 30, 'MaxIterations',
100, 'Display', 'off');
        [~, fval] = particleswarm(func_handle, D, lb, ub, options);
        results(i) = fval;
    end
    stats = [min(results), max(results), mean(results), std(results)];
end

function stats = run_sa(func_handle, D, lb, ub)
    results = zeros(15,1);
    for i = 1:15
        rng(i);
        x0 = lb + rand(1, D) .* (ub - lb);
        options = optimoptions('simulannealbnd', 'MaxIterations', 1000, 'Display',
'off');
        [~, fval] = simulannealbnd(func_handle, x0, lb, ub, options);
        results(i) = fval;
    end
    stats = [min(results), max(results), mean(results), std(results)];
end
```