Exercise -13

1) Implement Cohort Intelligence Algorithm (CIA) with the following parameters:
   a) Population size: 50
   b) Maximum number of iterations: 500

   Optimize any 5 test functions and provide the following:
   1. Mean function values for 20 runs
   2. Best function values for 20 runs
   3. Standard deviation values for 20 runs

Consider standard values for all other variables of the algorithm.

CODE:

```
% Define the test functions
sphere = @(x) sum(x.^2);
rastrigin = @(x) 10 * numel(x) + sum(x.^2 - 10 * cos(2 * pi * x));
rosenbrock = @(x) sum(100 * (x(2:end) - x(1:end-1).^2).^2 + (x(1:end-1) - 1).^2);
ackley = @(x) -20 * exp(-0.2 * sqrt(mean(x.^2))) - exp(mean(cos(2 * pi * x))) + 20 + exp(1);
griewank = @(x) 1 + sum(x.^2 / 4000) - prod(cos(x ./ sqrt(1:numel(x))));

% CIA Parameters
population_size = 50;
max_iterations = 500;
dimension = 10; % Adjust dimensionality if needed
bounds = [-5, 5]; % Bounds for the test functions

% Define the test function list
test_functions = {sphere, rastrigin, rosenbrock, ackley, griewank};
test_function_names = {'Sphere', 'Rastrigin', 'Rosenbrock', 'Ackley', 'Griewank'};

% Run CIA for each test function and collect statistics
results = struct();

for func_idx = 1:length(test_functions)
    func = test_functions{func_idx};
    func_name = test_function_names{func_idx};
```

```
best_values = zeros(1, 20);

for run = 1:20
    % Initialize population
    population = bounds(1) + (bounds(2) - bounds(1)) * rand(population_size, dimension);
    best_solution = inf;

    for iter = 1:max_iterations
        % Evaluate each individual's fitness
        fitness = arrayfun(@(idx) func(population(idx, :)), 1:population_size);

        % Track the best solution
        [min_fitness, min_idx] = min(fitness);
        if min_fitness < best_solution
            best_solution = min_fitness;
            best_position = population(min_idx, :);
        end

        % Update positions based on cohort behavior
        for i = 1:population_size
            if i ~= min_idx
                influence = (best_position - population(i, :)) .* rand(1, dimension);
                population(i, :) = population(i, :) + influence;
                population(i, :) = min(max(population(i, :), bounds(1)), bounds(2));
            end
        end
    end
    best_values(run) = best_solution;
end

% Calculate mean, best, and standard deviation values
mean_value = mean(best_values);
```

```
    best_value = min(best_values);
    std_dev = std(best_values);

    % Store results
    results.(func_name).MeanFunctionValue = mean_value;
    results.(func_name).BestFunctionValue = best_value;
    results.(func_name).StandardDeviation = std_dev;
end

% Display results
disp('CIA Optimization Results:');
for func_idx = 1:length(test_functions)
    func_name = test_function_names{func_idx};
    fprintf('Results for %s:\n', func_name);
    fprintf('Mean Function Value: %f\n', results.(func_name).MeanFunctionValue);
    fprintf('Best Function Value: %f\n', results.(func_name).BestFunctionValue);
    fprintf('Standard Deviation: %f\n\n', results.(func_name).StandardDeviation);
end
```

OUTPUT:

```
>> Assignment13
CIA Optimization Results:
Results for Sphere:
Mean Function Value: 1.439579
Best Function Value: 0.141891
Standard Deviation: 0.767616

Results for Rastrigin:
Mean Function Value: 29.557951
Best Function Value: 12.506783
Standard Deviation: 10.463805

Results for Rosenbrock:
Mean Function Value: 231.761684
Best Function Value: 39.335220
Standard Deviation: 186.761139

Results for Ackley:
Mean Function Value: 2.720634
Best Function Value: 1.269192
Standard Deviation: 0.723679

Results for Griewank:
Mean Function Value: 0.268737
Best Function Value: 0.038972
Standard Deviation: 0.162372
```