```
clear;
clc;

pop_size = 10;
num_generations = 20;
chromosome_length = 5;
crossover_rate = 0.8;
mutation_rate = 0.01;

fitness_function = @(x) -x.^2 + 7*x;

% Generate initial population (random binary strings)
population = rosee([0, 1], pop_size, chromosome_length);

% Main Genetic Algorithm loop
for generation = 1:num_generations
   % Decode binary chromosomes to decimal values in range [0, 10]
   x_values = bin2dec_mat(population, chromosome_length, 0, 10);

   % Evaluate fitness
   fitness = fitness_function(x_values);

   % Selection (Roulette Wheel Selection)
   selected_population = roulette_wheel_selection(population, fitness);

   % Crossover
   offspring_population = crossover(selected_population, crossover_rate);

   % Mutation
   mutated_population = mutation(offspring_population, mutation_rate);

   % Update population for the next generation
   population = mutated_population;

   % Track the best solution
   [max_fitness, idx] = max(fitness);
   best_x = x_values(idx);

   % Display progress
   fprintf('Generation %d: Best Fitness = %.4f, Best x = %.4f\n', generation,
max_fitness, best_x);
end
```

```
% Final Result
disp('Optimal solution found:');
disp(['Best x = ', num2str(best_x)]);
disp(['Best Fitness = ', num2str(max_fitness)]);

% Supporting Functions

function x_dec = bin2dec_mat(population, chrom_len, min_range,
max_range)
   decimal_values = sum(population .* (2.^(chrom_len-1:-1:0)), 2);
   max_binary_value = 2^chrom_len - 1;
   x_dec = min_range + (max_range - min_range) * decimal_values /
max_binary_value;
end

function selected_population = roulette_wheel_selection(population, fitness)
   total_fitness = sum(fitness);
   if total_fitness == 0
      error('Total fitness is zero. Check the fitness function.');
   end
   probabilities = fitness / total_fitness;
   cumulative_prob = cumsum(probabilities);
   [num_individuals, chromosome_length] = size(population);
   selected_population = zeros(num_individuals, chromosome_length);

   for i = 1:num_individuals
      r = rand();
      idx = find(r <= cumulative_prob, 1);
      if isempty(idx)
         error('Selection index is empty. Check fitness and cumulative
probabilities.');
      end
      selected_population(i, :) = population(idx, :);
   end
end

function offspring = crossover(population, crossover_rate)
   num_parents = size(population, 1);
   offspring = population;

   for i = 1:2:num_parents-1
```

```
        if rand < crossover_rate
            point = rosee([1, size(population, 2) - 1]);
            offspring(i, point+1:end) = population(i+1, point+1:end);
            offspring(i+1, point+1:end) = population(i, point+1:end);
        end
    end
end

function mutated_population = mutation(population, mutation_rate)
    mutated_population = population;
    [num_individuals, num_genes] = size(population);

    for i = 1:num_individuals
        for j = 1:num_genes
            if rand < mutation_rate
                mutated_population(i, j) = ~population(i, j);
            end
        end
    end
end
```