BIS LAB1 29/08/2025

Sharada Koundinya(1BM23CS310)

**CODE:**

```
import random
def fitness_function(x):
    return x ** 2

def decode(chromosome):
    return int(chromosome, 2)

def evaluate_population(population):
    return [fitness_function(decode(individual)) for individual in population]

def select(population, fitnesses):
    total_fitness = sum(fitnesses)
    if total_fitness == 0:
        return random.choice(population)
    pick = random.uniform(0, total_fitness)
    current = 0
    for individual, fitness in zip(population, fitnesses):
        current += fitness
        if current > pick:
            return individual

def crossover(parent1, parent2):
    if random.random() < CROSSOVER_RATE:
        point = random.randint(1, CHROMOSOME_LENGTH - 1)
        return (parent1[:point] + parent2[point:], parent2[:point] + parent1[point:])
    return parent1, parent2

def mutate(chromosome):
    new_chromosome = ''
    for bit in chromosome:
        if random.random() < MUTATION_RATE:
            new_chromosome += '0' if bit == '1' else '1'
        else:
            new_chromosome += bit
    return new_chromosome

def get_initial_population(size, length):
    population = []
    print(f"Enter {size} chromosomes (each of {length} bits, e.g., '10101'):")
```

```
    while len(population) < size:
        chrom = input(f"Chromosome {len(population)+1}: ").strip()
        if len(chrom) == length and all(bit in '01' for bit in chrom):
            population.append(chrom)
        else:
            print(f"Invalid input. Please enter a {length}-bit binary string.")
    return population

def genetic_algorithm():
    population = get_initial_population(POPULATION_SIZE, CHROMOSOME_LENGTH)
    best_solution = None
    best_fitness = float('-inf')

    for generation in range(GENERATIONS):
        fitnesses = evaluate_population(population)

        for i, individual in enumerate(population):
            if fitnesses[i] > best_fitness:
                best_fitness = fitnesses[i]
                best_solution = individual

        print(f"Generation {generation + 1}: Best Fitness = {best_fitness}, Best x =
{decode(best_solution)}")

        new_population = []
        while len(new_population) < POPULATION_SIZE:
            parent1 = select(population, fitnesses)
            parent2 = select(population, fitnesses)
            offspring1, offspring2 = crossover(parent1, parent2)
            offspring1 = mutate(offspring1)
            offspring2 = mutate(offspring2)
            new_population.extend([offspring1, offspring2])

        population = new_population[:POPULATION_SIZE]

    print("\nBest solution found:")
    print(f"Chromosome: {best_solution}")
    print(f"x = {decode(best_solution)}")
    print(f"f(x) = {fitness_function(decode(best_solution))}")

POPULATION_SIZE = 4
CHROMOSOME_LENGTH = 5
MUTATION_RATE = 0.01
```

```
CROSSOVER_RATE = 0.8
GENERATIONS = 20

if __name__ == "__main__":
    genetic_algorithm()
```

**OUTPUT:**

```
Enter 4 chromosomes (each of 5 bits, e.g., '10101'):
Chromosome 1: 01100
Chromosome 2: 11001
Chromosome 3: 00101
Chromosome 4: 10011
Generation 1: Best Fitness = 625, Best x = 25
Generation 2: Best Fitness = 625, Best x = 25
Generation 3: Best Fitness = 625, Best x = 25
Generation 4: Best Fitness = 625, Best x = 25
Generation 5: Best Fitness = 625, Best x = 25
Generation 6: Best Fitness = 625, Best x = 25
Generation 7: Best Fitness = 625, Best x = 25
Generation 8: Best Fitness = 625, Best x = 25
Generation 9: Best Fitness = 625, Best x = 25
Generation 10: Best Fitness = 625, Best x = 25
Generation 11: Best Fitness = 625, Best x = 25
Generation 12: Best Fitness = 625, Best x = 25
Generation 13: Best Fitness = 625, Best x = 25
Generation 14: Best Fitness = 625, Best x = 25
Generation 15: Best Fitness = 625, Best x = 25
Generation 16: Best Fitness = 625, Best x = 25
Generation 17: Best Fitness = 625, Best x = 25
Generation 18: Best Fitness = 625, Best x = 25
Generation 19: Best Fitness = 625, Best x = 25
Generation 20: Best Fitness = 625, Best x = 25

Best solution found:
Chromosome: 11001
x = 25
f(x) = 625
```