```python
import random


def objective_function(x):
    return x ** 2

POPULATION_SIZE = 20
MUTATION_RATE = 0.01
CROSSOVER_RATE = 0.7
NUM_GENERATIONS = 50
GENOME_LENGTH = 5


def create_population(size):
    return [random.randint(0, 31) for _ in range(size)]


def evaluate_fitness(population):
    return [objective_function(ind) for ind in population]


def select(population, fitness):

    return random.choices(population, weights=fitness, k=2)


def crossover(parent1, parent2):
    if random.random() < CROSSOVER_RATE:
        point = random.randint(1, GENOME_LENGTH - 1)
        mask = (1 << point) - 1
        offspring1 = (parent1 & mask) | (parent2 & ~mask)
        offspring2 = (parent2 & mask) | (parent1 & ~mask)
        return offspring1, offspring2
    else:
        return parent1, parent2


def mutate(individual):
    if random.random() < MUTATION_RATE:
        mutation_point = random.randint(0, GENOME_LENGTH - 1)
        individual ^= (1 << mutation_point)
    return individual


def genetic_algorithm():
    population = create_population(POPULATION_SIZE)
    best_solution = None
    best_fitness = -1

    for generation in range(NUM_GENERATIONS):
```

```
        fitness = evaluate_fitness(population)
        new_population = []


        current_best = max(fitness)
        if current_best > best_fitness:
            best_fitness = current_best
            best_solution = population[fitness.index(current_best)]


        for _ in range(POPULATION_SIZE // 2):
            parent1, parent2 = select(population, fitness)
            offspring1, offspring2 = crossover(parent1, parent2)
            new_population.append(mutate(offspring1))
            new_population.append(mutate(offspring2))

        population = new_population

    return best_solution, best_fitness

best_solution, best_fitness = genetic_algorithm()
print(f'Best Solution: x = {best_solution}, f(x) = {best_fitness}')
```

output:-