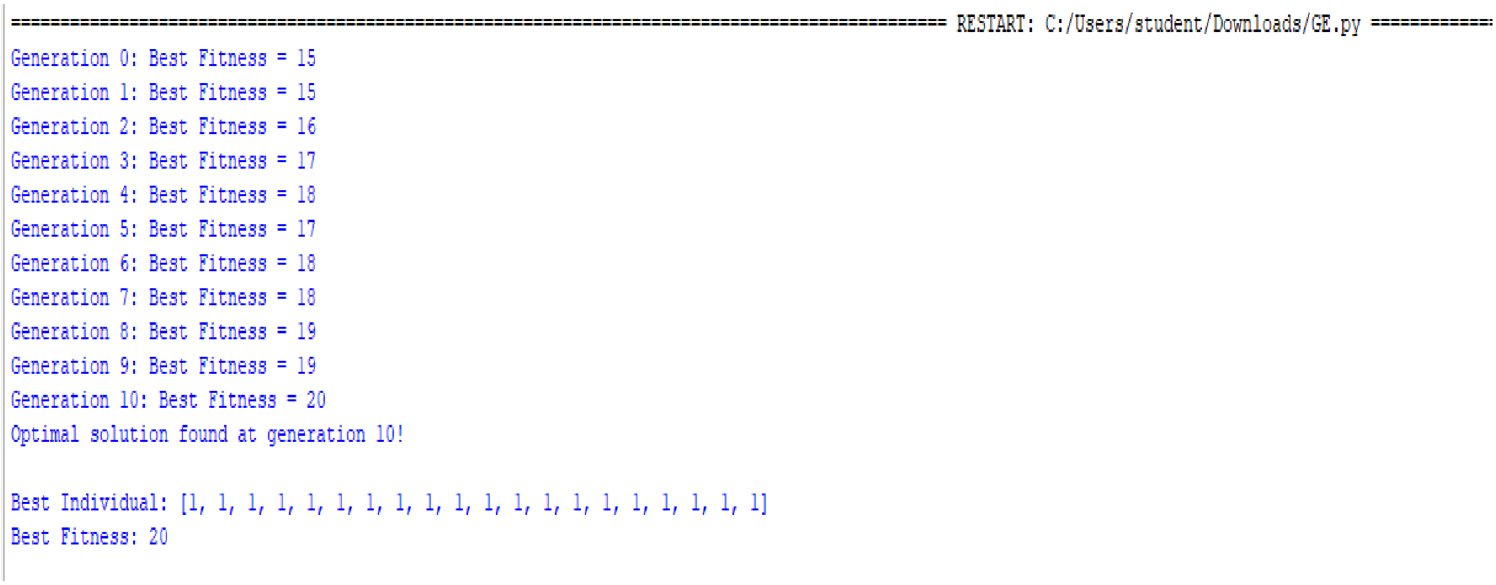
GENETIC ALGORITHM

CODE:-

```
import random


POP_SIZE = 100
CHROM_LENGTH = 20
MAX_GEN = 100
MUTATION_RATE = 0.01


random.seed(42)


class Individual:
    def __init__(self, genes=None):
        self.genes = genes if genes is not None else [random.randint(0, 1) for _ in
range(CHROM_LENGTH)]
        self.fitness = self.evaluate_fitness()

    def evaluate_fitness(self):
        return sum(self.genes)



def initialize_population():
    return [Individual() for _ in range(POP_SIZE)]

def select_parent(population):
    i, j = random.sample(range(POP_SIZE), 2)
    return population[i] if population[i].fitness > population[j].fitness else population[j]

def crossover(parent1, parent2):
    point = random.randint(1, CHROM_LENGTH - 1)  # Avoid 0 to prevent cloning
    child_genes = parent1.genes[:point] + parent2.genes[point:]
    return Individual(child_genes)

def mutate(individual):
    for i in range(CHROM_LENGTH):
        if random.random() < MUTATION_RATE:
            individual.genes[i] = 1 - individual.genes[i]
    individual.fitness = individual.evaluate_fitness()
```

```
def get_best_individual(population):
    return max(population, key=lambda ind: ind.fitness)


def genetic_algorithm():
    population = initialize_population()

    for gen in range(MAX_GEN):
        new_population = []
        for _ in range(POP_SIZE):
            parent1 = select_parent(population)
            parent2 = select_parent(population)
            child = crossover(parent1, parent2)
            mutate(child)
            new_population.append(child)

        population = new_population
        best = get_best_individual(population)

        print(f"Generation {gen}: Best Fitness = {best.fitness}")

        if best.fitness == CHROM_LENGTH:
            print(f"Optimal solution found at generation {gen}!")
            break

    best = get_best_individual(population)
    print(f"\nBest Individual: {best.genes}")
    print(f"Best Fitness: {best.fitness}")


genetic_algorithm()
```

OUTPUT:-



```
=============================================================================================== RESTART: C:/Users/student/Downloads/GE.py ============
Generation 0: Best Fitness = 15
Generation 1: Best Fitness = 15
Generation 2: Best Fitness = 16
Generation 3: Best Fitness = 17
Generation 4: Best Fitness = 18
Generation 5: Best Fitness = 17
Generation 6: Best Fitness = 18
Generation 7: Best Fitness = 18
Generation 8: Best Fitness = 19
Generation 9: Best Fitness = 19
Generation 10: Best Fitness = 20
Optimal solution found at generation 10!

Best Individual: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Best Fitness: 20
```