LAB # 13

Implementing Metaheuristic Algorithm (Genetic Algorithm)

Lab Tasks:-

- Run the given code of Genetic Algorithm (Coin Toss Problem) and show the output.

Code:- **Output:-**

```python
# TASK 1: Genetic Algorithm – Coin Toss Problem

def fitness(chromosome):
    return sum(chromosome)

# Hardcoded chromosomes
population = [
    [1, 0, 1, 1, 0, 1, 0, 1],
    [0, 1, 1, 0, 1, 0, 1, 1],
    [1, 1, 0, 0, 1, 1, 0, 0],
    [0, 0, 1, 1, 1, 0, 1, 0],
    [1, 0, 0, 1, 0, 1, 1, 0],
    [0, 1, 0, 1, 1, 0, 0, 1]
]

print("Initial Population and Fitness:\n")
for i, chrom in enumerate(population):
    print(f"S{i+1}: {chrom} Fitness = {fitness(chrom)}")
```

```
Initial Population and Fitness:

S1: [1, 0, 1, 1, 0, 1, 0, 1] Fitness = 5
S2: [0, 1, 1, 0, 1, 0, 1, 1] Fitness = 5
S3: [1, 1, 0, 0, 1, 1, 0, 0] Fitness = 4
S4: [0, 0, 1, 1, 1, 0, 1, 0] Fitness = 4
S5: [1, 0, 0, 1, 0, 1, 1, 0] Fitness = 4
S6: [0, 1, 0, 1, 1, 0, 0, 1] Fitness = 4
```

- In given code there are crossover functions implemented between S1, S4 and S5, S6 after 3rd point. You can perform crossover between S2 and S3 after 4th point.

Code:- **Output:-**

```python
def crossover(parent1, parent2, point):
    child1 = parent1[:point] + parent2[point:]
    child2 = parent2[:point] + parent1[point:]
    return child1, child2

population = [
    [1, 0, 1, 1, 0, 1, 0, 1],  # S1
    [0, 1, 1, 0, 1, 0, 1, 1],  # S2
    [1, 1, 0, 0, 1, 1, 0, 0],  # S3
    [0, 0, 1, 1, 1, 0, 1, 0],  # S4
    [1, 0, 0, 1, 0, 1, 1, 0],  # S5
    [0, 1, 0, 1, 1, 0, 0, 1]   # S6
]
population[0], population[3] = crossover(population[0], population[3], 3)

population[4], population[5] = crossover(population[4], population[5], 3)

population[1], population[2] = crossover(population[1], population[2], 4)

print("Population After Crossover:\n")
for i, chrom in enumerate(population):
    print(f"S{i+1}: {chrom}")
```

```
Population After Crossover:

S1: [1, 0, 1, 1, 1, 0, 1, 0]
S2: [0, 1, 1, 0, 1, 1, 0, 0]
S3: [1, 1, 0, 0, 1, 0, 1, 1]
S4: [0, 0, 1, 1, 0, 1, 0, 1]
S5: [1, 0, 0, 1, 1, 0, 0, 1]
S6: [0, 1, 0, 1, 0, 1, 1, 0]
```

- In given code the values of chromosomes are hardcoded, you may take input values of chromosomes at run time.

Code:- **Output:-**

```python
# TASK 3: Runtime Input of Chromosomes

def fitness(chromosome):
    return sum(chromosome)

population = []

n = int(input("Enter number of chromosomes: "))
length = int(input("Enter chromosome length: "))

print("\nEnter chromosomes (0 for T, 1 for H):")
for i in range(n):
    chrom = list(map(int, input(f"S{i+1}: ").split()))
    population.append(chrom)

print("\nChromosomes and Fitness:\n")
for i, chrom in enumerate(population):
    print(f"S{i+1}: {chrom} Fitness = {fitness(chrom)}")
```

```
Enter number of chromosomes:  3
Enter chromosome length:  5

Enter chromosomes (0 for T, 1 for H):
S1:  0
S2:  1
S3:  0

Chromosomes and Fitness:

S1: [0] Fitness = 0
S2: [1] Fitness = 1
S3: [0] Fitness = 0
```