Q1. Explain each step of the genetic algorithm using the **Traveling Salesman Problem (TSP)** as the application scenario. Clearly describe population initialization, fitness evaluation, selection, crossover, and mutation using appropriate terminology. Illustrate your explanation with an example involving at least five cities and provide step-by-step calculations wherever applicable."

Q2. Consider the function of maximizing the function $f(x) = x^2$, where x is permitted to vary between 0 to 31 using Genetic Algorithm.

Q3. Maximize the value of the function $f(x) = -x\text{^}2 + 2x$ over the range of real number from 0 to 2 with initial population 11010, 00111, 10110, 00101, with random number 0.4, 0.15, 0.7, 0.9. Select the crossover between the first and fifth digits?

Q4. Solve a problem $f(x) = x^3$, where $0 \leq x \leq 15$. Find some stages of application of genetic algorithms.

Q5. Suppose a genetic algorithm uses chromosomes of the form $x = abcdefgh$ with a fixed length of eight genes. Each gene can be any digit between 0 and 9. Let the fitness of individual x be calculated as:

$$f(x) = (a + b) - (c + d) + (e + f) - (g + h)$$

and let the initial population consist of four individuals with the following chromosomes:

$$\boldsymbol{x1} = \mathbf{6\,5\,4\,1\,3\,5\,3\,2} \quad , \quad \boldsymbol{x2} = \mathbf{8\,7\,1\,2\,6\,6\,0\,1}$$

$$\boldsymbol{x3} = \mathbf{2\,3\,9\,2\,1\,2\,8\,5} \quad , \quad \boldsymbol{x4} = \mathbf{4\,1\,8\,5\,2\,0\,9\,4}$$

Evaluate each individual's fitness, showing all your workings and arrange them in order with the fittest first and the least fit last.

Q1. Answer: Genetic Algorithm (GA) is a heuristic search technique inspired by Charles Darwin's theory of natural evolution. It is widely used to solve optimization and search problems.

When applied to the **Traveling Salesman Problem (TSP)**, the GA attempts to find the shortest possible route that visits each city exactly once and returns to the starting city.

Steps to solve TSP using Genetic Algorithm:

i. Define Encoding Mechanism (Chromosome Representation)

- In TSP, an **individual (chromosome)** is a possible tour (i.e., an ordered list of cities).
- For example, for **5 cities A, B, C, D, E**, one possible chromosome is:
- Chromosome = [A, C, E, D, B]

ii. Define Fitness Function

- The goal is to **minimize the total tour length**.
- Fitness function:
- $$\text{Fitness} = \frac{1}{\text{Total distance of the Tour}}$$
- This ensures shorter routes have higher fitness values.

iii. Initialize a Population (Assume population size = 4)

Let the distances (in km) between cities be defined by the following symmetric matrix:

	A	B	C	D	E
A	0	10	15	20	25
B	10	0	35	25	30
C	15	35	0	30	20
D	20	25	30	0	15
E	25	30	20	15	0

Initial population:

Parent	**Chromosome Route**	**Parent**	**Chromosome Route**
P1	A → B → C → D → E → A	P3	A → D → C → B → E → A
P2	A → C → E → B → D → A	P4	A → E → D → C → B → A

iv. Calculate the Fitness of all individuals

Compute the tour length and fitness:

- **P1**: A-B-C-D-E-A = 10+35+30+15+25 = **115**, Fitness = 1/115 ≈ **0.0087**
- **P2**: A-C-E-B-D-A = 15+20+30+25+20 = **110**, Fitness = 1/110 ≈ **0.0091**
- **P3**: A-D-C-B-E-A = 20+30+35+30+25 = **140**, Fitness = 1/140 ≈ **0.0071**
- **P4**: A-E-D-C-B-A = 25+15+30+35+10 = **115**, Fitness = 1/115 ≈ **0.0087**

v. Selection of Parents

- Use **roulette wheel selection** based on fitness.
- Assign probabilities:
    - P1: 0.0087 / total ≈ 0.22
    - P2: 0.0091 / total ≈ 0.23
    - P3: 0.0071 / total ≈ 0.18
    - P4: 0.0087 / total ≈ 0.22

Select **P2** and **P4** for crossover (based on higher fitness).

vi. Crossover

- Use **Order Crossover (OX)** for permutation-based chromosomes.
- Example:
    - **Parent 1**: A-C-E-B-D
    - **Parent 2**: A-E-D-C-B
    - Crossover between position 3 to 4:
        - Segment from P1: **E-B**
        - Fill remaining from P2 preserving order: A-D-C
        - Resultant child

          C1: [_, _, E, B, _]   =>   [A, D, E, B, C]

          C2: [_ , _ , D, C, _] = > [A, E, D, C, B]

vii. Mutation

- Apply **swap mutation** to introduce diversity.
- E.g., swap position 2 and 4:
    - **Before C1**: [A, D, E, B, C]
    - After: **C3** [A, B, E, D, C]

viii. Stopping Criteria

- Continue repeating steps **iv to vii** until:
    - A solution with minimum tour distance is found,
    - or max generations reached,
    - or improvement plateaus.

Conclusion: Using GA on TSP helps approximate near-optimal routes efficiently. The solution evolves generation-by-generation through selection, crossover, and mutation — mimicking natural evolution to solve combinatorial optimization problems.

Q1 More explaining about stopping conditions

✅ **Common Stopping Criteria for GA in TSP:**

1. Maximum Number of Generations

- 🔧 Stop after a fixed number of generations (e.g., 500, 1000).
- ✅ Simple and ensures time-bounded execution.
- ❌ Might stop too early or run unnecessarily long.

2. No Improvement in Best Fitness

- 🕵 Monitor the best route length over time.
- ⏸ Stop if no improvement occurs for N generations.
- Example: "If best distance hasn't changed for 100 generations, stop."
- ✅ Efficient and adaptive.
- ❌ Might stop prematurely due to local optima.

3. Target Fitness Reached

- 🎯 Stop when a route below a desired length is found.
- Example: If optimal/near-optimal TSP length is known, set a threshold.
- ✅ Works well if you know the benchmark.
- ❌ Not always feasible; real-world TSP solutions may not have a known best.

4. Time Limit

- ⏲ Stop after a fixed amount of computation time.
- ✅ Useful in real-time or time-constrained systems.
- ❌ Doesn't guarantee best possible solution.

5. Diversity Threshold

- 🧬 Measure how similar the population has become.
- 🛑 Stop when most individuals in the population are too similar.

- ✅ Indicates convergence (or stagnation).
- ❌ Can be computationally expensive to track.

6. Combination of Criteria ✅ (Recommended)

- Use multiple conditions for better control:

Stop if:

- Max generations reached, OR

- No improvement in best fitness for 100 generations, OR

- Time limit exceeded

🧠 **Best Practice for TSP:**

- TSP is a hard NP-Hard problem.
- It's better to balance exploration and exploitation, so using:

✅ **"No improvement in N generations" + "Max generations"**

is usually the most effective combo.