CL-3 VIVA

PRAC – 1

What is RPC?

RPC (Remote Procedure Call) is a communication protocol that allows a program to call (invoke) a function or procedure **on another computer (server)** as if it were a local function.

In Simple Terms:

RPC lets one machine **request a service** (like a function call) from another machine **without worrying about how the communication happens**.

What is CI and DC??

PRAC – 2

Pyro4 is Python remote object

What is RMI??

RMI (Remote Method Invocation) is a **Java-specific** mechanism that allows an object on one Java Virtual Machine (JVM) to **invoke methods on an object running in another JVM**, typically over a network.
But we have used Pyro4 library , a powerful library for implementing **RPC-like behavior in Python**

What is the purpose of @Pyro4.expose?

The purpose of @Pyro4.expose is to **explicitly mark a class or method as accessible remotely** by Pyro clients. Without @Pyro4.expose, Pyro will not allow clients to call that method over the network. Only methods explicitly marked with @expose can be accessed remotely

What is URI??

A **URI (Uniform Resource Identifier)** is a unique address that identifies a **resource** — in this case, the **remote object on the Pyro server** — so the client knows where to find and connect to it.

The URI looks something like this:

PYRO:obj_abc123@localhost:55555

it contains:

1. **Protocol**: PYRO – specifies it's a Pyro object.
2. **Object ID**: obj_abc123 – the registered name of the object.
3. **Host**: localhost – where the server is running.
4. **Port**: 55555 – the port number the server is listening on.

Is Pyro4 synchronous or asynchronous?

Pyro4 is synchronous by default.

This means:

- When the client calls a remote method, it **waits (blocks)** until the server finishes processing and sends back the result.

PRAC-3

What is Hadoop ?

Hadoop is an **open-source framework** developed by Apache for **storing** and **processing large amounts of data** in a **distributed** and **scalable** manner using **clusters of computers**.

What is MRJob?

A Python framework to write MapReduce jobs that can run locally or on Hadoop.

What is Map Reduce?

MapReduce is a **programming model** used for processing and generating large datasets in a **parallel** and **distributed** manner. It divides the task into two main steps: **Map** and **Reduce**.

Map Phase:

- The input data is **split into smaller chunks** and processed in parallel.
- Each chunk is processed by a **mapper** which transforms it into a set of key-value pair

Shuffle & Sort:

- After the map phase, the data is **grouped** by keys, and the values for each key are sorted and sent to the **reduce phase**.

Reduce Phase:

- The **reducer** aggregates the values for each key (e.g., summing counts, finding max values).
- The output is the **final result** after processing.

What is RegEx?

Regex (Regular Expression) is a **pattern** used to match **character strings** in text

What is the yield Keyword in Python?

The yield keyword is used in a function to make it a **generator**. When a function contains yield, it doesn't return a single value but instead **yields** a series of values one at a time, pausing and resuming its execution each time.

PRAC- 4

What is Load Balancing?

Load balancing is the process of distributing incoming network or application traffic across multiple servers or resources to ensure no single server becomes overloaded. By balancing the load efficiently, the system improves performance, scalability, and reliability, preventing any single point of failure.

Random Selection:

- **Concept**: In this method, the load balancer randomly selects a server from the pool of available servers to handle each incoming request.

- **Advantages**:
  - Simple and easy to implement.
  - Useful when servers have roughly the same capacity and the incoming traffic is random or has no significant pattern.

- **Disadvantages**:
  - May lead to uneven load distribution, as some servers might get more requests than others purely by chance.

Round Robin:

- **Concept**: The load balancer assigns requests to servers in a cyclic manner. After the last server is chosen, the load balancer starts over with the first server.
- **How it works**:
  - Servers are listed in a queue.
  - The load balancer assigns the first request to the first server, the second request to the second server, and so on.
  - Once it reaches the last server, it starts again with the first one.

 Advantages:

- Easy to implement and ensures that each server gets an equal number of requests.
- Works well when all servers have similar capacity and are capable of handling similar loads.

 Disadvantages:

- Does not account for the load or number of active connections on each server, so servers with heavier loads may still receive more requests than others.

Least Connections:

- **Concept**: This algorithm routes the request to the server that currently has the least number of active connections or requests being handled.

How it works:

- The load balancer checks the number of active connections or tasks on each server.
- It assigns the next incoming request to the server with the least load (fewest active connections).

 Advantages:

- More efficient, especially when servers have varying processing capacities or the load is not uniform.
- Helps ensure that servers are not overloaded, as the load balancer keeps an eye on the current active connections.

 Disadvantages:

- May lead to uneven load distribution if some servers are consistently underutilized or overutilized due to factors like long-running connections.

PRAC – 5

What is Clonal Selection Algorithm?

The **Clonal Selection Algorithm (CSA)** is an optimization and machine learning algorithm inspired by the **clonal selection principle** of the **biological immune system**. It's mainly used in **Artificial Immune Systems (AIS)** to solve optimization, pattern recognition, and anomaly detection problems.

In biology, **clonal selection** is a process where immune cells (like B-cells) that recognize a pathogen are selected to proliferate (clone) and mutate to better respond to the invader. The immune system then keeps the best-fitting clones (antibodies) to fight the infection.

How Clonal Selection Algorithm Works:

1. **Generate initial population**:
    - Randomly create a population of candidate solutions (called **antibodies**).
2. **Evaluate fitness**:

    - Assess how well each antibody solves the given problem using a **fitness function**.
3. **Select best antibodies**:
    - Choose the top-performing antibodies based on fitness (analogous to selecting the strongest immune cells).
4. **Clone the best antibodies**:
    - Create multiple copies (clones) of the selected antibodies. More clones for better antibodies.
5. **Hypermutation**:
    - Mutate the clones slightly to introduce diversity.
    - Better antibodies mutate less, and worse ones mutate more.
6. **Re-select the best**:
    - Evaluate the mutated clones and keep the best ones.
7. **Replace some of the population**:
    - Optionally replace a few weak solutions with new random ones to maintain diversity.
8. **Repeat**:
    - The process continues for a fixed number of iterations or until an optimal solution is found.

PRAC – 6

What is Neural Style Transfer (NST)?

Neural Style Transfer is a deep learning technique that **blends two images**:

- A **content image** (e.g., your photo),
- A **style image** (e.g., a painting by Van Gogh),

...to produce a **new image** that **retains the content** of the first but **adopts the artistic style** of the second.

Applications:

- Artistic filters
- Augmented reality
- Generative art
- Style transformation in video

PRAC – 7

What is AIS?

AIS stands for **Artificial Immune System**. It is a branch of artificial intelligence inspired by the **biological immune system**. The biological immune system is responsible for recognizing and defending against foreign invaders (such as viruses and bacteria). AIS mimics this process to solve complex problems, primarily in **pattern recognition**, **optimization**, and **classification** tasks.

Key Concepts of AIS:

1. **Memory Cells (Antibodies)**:
    - In the biological immune system, **antibodies** recognize and attack pathogens (foreign invaders). In AIS, **memory cells** represent solutions or patterns learned from past data.
2. **Clonal Selection**:
    - When an immune system detects a pathogen, it produces **clones** of the most effective antibodies to target it more efficiently. Similarly, AIS uses **clonal selection** to find the best solutions to problems by cloning and modifying (mutating) high-quality solutions.
3. **Affinity-based Selection**:
    - Antibodies are selected based on their **affinity** to the target (i.e., how well they match the problem). AIS applies this concept to select the best solutions based on how closely they match the desired outcome.

4. **Mutation**:
    - After cloning, small random changes (mutations) are introduced to the clones to explore new possible solutions and avoid getting stuck in suboptimal solutions (local optima).

AIS Applications:

1. **Pattern Recognition**: Identifying patterns in data (e.g., classification tasks like damage detection).
2. **Optimization**: Solving complex optimization problems like function optimization, scheduling, and routing problems.
3. **Anomaly Detection**: Identifying abnormal patterns, outliers, or faults in systems.
4. **Classification**: Assigning labels to data points based on past experiences (e.g., the structure damage classification you mentioned).

PRAC – 8

What is DEAP?

DEAP (Distributed Evolutionary Algorithms in Python) is a **flexible evolutionary computation framework** written in **Python**. It allows researchers and developers to easily create and evaluate **evolutionary algorithms (EAs)**, such as:

- Genetic Algorithms (GA)
- Genetic Programming (GP)
- Evolution Strategies (ES)
- Multi-objective Optimization (e.g., NSGA-II)

Applications:

- Optimization problems (engineering, economics)
- Automated design
- Machine learning hyperparameter tuning
- Game strategies
- Robotics and control systems

PRAC 10

What is ACO?

Ant Colony Optimization (ACO) is a **nature-inspired optimization algorithm** based on the **foraging behavior of ants**. It's especially effective for solving **combinatorial optimization problems** like:

- Shortest path problems (e.g., Traveling Salesman Problem)
- Scheduling
- Routing (e.g., network and logistics optimization)

Inspiration:

Real ants find the shortest path between their colony and food by:

1. Randomly exploring paths.
2. Depositing **pheromones** on the paths they travel.
3. Following stronger pheromone trails (i.e., better/shorter paths).
4. Over time, **shorter paths have stronger pheromone concentrations**, attracting more ants and reinforcing those routes.

Use Cases:

- Traveling Salesman Problem (TSP)
- Network routing
- Vehicle routing
- Job scheduling
- Feature selection in ML