Day 25: Learning Summary

1. Aptitude – Inequalities

Inequality-based reasoning involves comparing expressions using symbols: `>`, `<`, `>=`, `<=`, and `=`.

Key Concepts:

- **Definite vs. Possible:** A conclusion is only true if it holds in all possible scenarios derived from the statements.
- **Range Analysis:** Identifying the overlap between multiple inequalities (e.g., if $x > 5$ and $x < 10$, then $5 < x < 10$).
- **The "Either-Or" Case:** Occurs when two conclusions are individually false but combined cover all possibilities (e.g., $A > B$ and $A = B$ when the statement says $A \geq B$).
- **Coded Inequalities:** Symbols like $@, \#, \$$ are used to represent comparison operators. Solving requires decoding them first.

2. Programming – Longest Palindromic Substring

Finding the longest substring within a string that reads the same forwards and backwards.

Approach: Expand Around Center

1. **Iterate:** Loop through each character in the string.
2. **Symmetry Check:** Treat each character (and the gap between characters) as a potential center.
3. **Expand:** Move outward from the center as long as the characters match.
    - *Odd length:* Center is one character (e.g., "aba").
    - *Even length:* Center is between two characters (e.g., "abba").
4. **Track:** Keep track of the maximum length and the starting index found so far.

Complexity: $O(n^2)$ time and $O(1)$ space.

3. Concept – Python Regex (`re` module)

Regular Expressions (Regex) provide a powerful way to search and manipulate text patterns.

Essential Functions:

- `re.search()` : Scans a string for a match to the pattern.
- `re.findall()` : Returns a list containing all matches.
- `re.sub()` : Replaces one or many matches with a string.

Common Meta-characters:

- `^` : Starts with.
- `$` : Ends with.
- `.` : Any character.
- `\d` : Matches any digit (0-9).
- `+` : One or more occurrences.
- `*` : Zero or more occurrences.

4. C/C++ Concept – Pattern Matching

At a low level, pattern matching involves comparing sequences of characters to find a "needle" in a "haystack."

Key Algorithms:

- **Naive Approach:** Check every possible position in the text for the pattern. $O(m \times n)$.
- **KMP (Knuth-Morris-Pratt):** Uses a prefix table to avoid redundant comparisons by skipping characters that have already been matched.
- **Boyer-Moore:** Matches from right to left and uses "bad character" shifts to jump ahead.

5. SQL – VIEWS

A **View** is a virtual table based on the result-set of an SQL statement. It does not store data itself but acts as a "saved query."

Why Use Views?

- **Simplification:** Hides complex joins or calculations. Instead of writing a 10-line query every time, you just query the view.
- **Security:** You can give a user access to a View containing only specific columns without giving them access to the underlying base tables.
- **Consistency:** Ensures that everyone is looking at the same derived data logic.

Syntax:

```
-- Creating a View
CREATE VIEW employee_summary AS
SELECT name, department, salary
FROM employees
WHERE active = 1;

-- Using a View
SELECT * FROM employee_summary;

-- Deleting a View
```

```
DROP VIEW employee_summary;
```