📚 Day 15 Study Notes

1. Aptitude: Mixtures & Alligations

Goal: Solve problems involving mixing two ingredients with different prices or concentrations to obtain a mixture of a desired price/concentration.

The Rule of Alligation

This rule helps calculate the ratio in which two ingredients must be mixed.

Formula:

$$\frac{\text{Quantity of Cheaper}}{\text{Quantity of Dearer}} = \frac{\text{Price of Dearer } (d) - \text{Mean Price } (m)}{\text{Mean Price } (m) - \text{Price of Cheaper } (c)}$$

Visual Representation:

```
    Cheaper Price (c)          Dearer Price (d)
                 \            /
                  \          /
                 Mean Price (m)
                  /          \
                 /            \
      (d - m)                  (m - c)
```

- **Ratio:** $(d - m) : (m - c)$

Example Problem

Question: In what ratio must rice at $10/kg be mixed with rice at $20/kg to get a mixture worth $14/kg?

- Cheaper (c) = 10
- Dearer (d) = 20
- Mean (m) = 14
- Ratio = $(20 - 14) : (14 - 10) = 6 : 4 = 3 : 2$
- **Answer:** Mix 3 parts of cheaper rice with 2 parts of dearer rice.

2. Programming: Linear Search

Goal: Find an element in a list by checking every element sequentially.

Logic

1. Start from the first element (index 0).

2. Compare the current element with the target value.
3. If they match, return the current index.
4. If the loop finishes without finding a match, return -1 (or indicate not found).

Complexity

- **Time Complexity:** $O(N)$ (Worst case: element is at the end or not present).
- **Space Complexity:** $O(1)$ (No extra space needed).

Code Snippet (Python)

```python
def linear_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i  # Return the index where found
    return -1  # Return -1 if not found

numbers = [10, 50, 30, 70, 80, 20]
key = 30
index = linear_search(numbers, key)
print(f"Element found at index: {index}")
```

3. Concept: Python Constructors

Goal: Initialize objects automatically when they are created.

The `__init__` Method

- In Python, the constructor is a special method named `__init__` .
- It is automatically called when a new instance (object) of a class is created.
- It is used to assign values to object properties or perform setup steps.

Example

```python
class Student:
    # Constructor
    def __init__(self, name, age):
        self.name = name  # Initializing attributes
        self.age = age

    def display(self):
        print(f"Student: {self.name}, Age: {self.age}")

# The constructor is called here automatically
s1 = Student("John", 21)
s1.display()
```

4. C/C++ Concept: Constructors

Goal: Understand object initialization in statically typed languages.

Key Characteristics

1. **Name:** Must have the **same name** as the class.
2. **Return Type:** Does **not** have a return type (not even `void`).
3. **Automatic Call:** Called automatically when an object is instantiated.

Types of Constructors

- **Default Constructor:** Has no parameters.
- **Parameterized Constructor:** Accepts arguments to initialize data members.

Example (C++)

```cpp
#include <iostream>
using namespace std;

class Point {
public:
    int x, y;

    // Parameterized Constructor
    Point(int valX, int valY) {
        x = valX;
        y = valY;
    }
};

int main() {
    Point p1(10, 20); // Constructor called here
    cout << "X: " << p1.x << ", Y: " << p1.y;
    return 0;
}
```

5. SQL: DELETE Statement

Goal: Remove existing records from a table.

Syntax

```sql
DELETE FROM table_name
WHERE condition;
```

⚠️ Critical Warning (The "WHERE" Clause)

- If you omit the `WHERE` clause, **ALL** records will be deleted! The table structure will remain, but it will be empty.
- **Difference from `DROP`:** `DELETE` removes data row by row (and can be rolled back in transactions). `DROP` deletes the table structure entirely.
- **Difference from `TRUNCATE`:** `TRUNCATE` wipes all data faster but cannot be filtered with `WHERE`.

Examples

1. Delete a specific record: Remove the employee with ID 5.

```
DELETE FROM Employees
WHERE EmpID = 5;
```

2. Delete based on a condition: Remove all orders placed before 2020.

```
DELETE FROM Orders
WHERE OrderDate < '2020-01-01';
```