

**Design Document: Employee Stock Ownership Plan (ESOP) Eligibility System using Divide-and-Conquer**

**1. Overview**

This document outlines the design of a system that determines employee eligibility for Employee Stock Ownership Plans (ESOPs) based on specific criteria. The system categorizes employee IDs into different groups and calculates eligibility for extra ESOPs.

**2. Problem Statement**

Given a set of 5-digit employee IDs (ranging from 10000 to 99999), the system must:

* Categorize IDs into three eligibility groups:
  + Group 1: Palindromic IDs.
  + Group 2: IDs divisible by 25.
  + Group 3: IDs whose digit sum is divisible by 5.
* Identify IDs eligible for an additional 10% ESOP allocation, defined as those belonging to both Group 1 and Group 3.
* Generate a comprehensive report detailing eligibility statistics and selected ID samples

**3. Design Considerations -**

**3.1. Algorithm: Divide-and-Conquer**

* The process\_employee\_ids() function employs a recursive divide-and-conquer strategy.
* The input list of employee IDs is recursively partitioned into sub-lists until each sub-list contains a single ID.
* The determine\_employee\_eligibility() function is applied to each individual ID, generating a dictionary of eligibility group memberships.
* The results from the sub-lists are then merged, forming a consolidated eligibility dictionary.

**3.2. Data Structures**

* Input: A text file (inputPS07.txt) containing employee IDs, one per line.
* Output: A text file (outputPS07.txt) containing the report.
* Eligibility Dictionary: A Python dictionary (eligibility\_dict) is used to store the eligibility status of each employee ID. Keys are employee IDs, and values are lists of corresponding eligibility group strings.
* Group Lists: Python lists (palindrome\_group, divisible\_by\_25\_group, sum\_divisible\_by\_5\_group, extra\_esop\_group) are used to store the employee ids that are members of each respective group.

**3.3. Function Overview**

* is\_number\_palindrome(number): Determines if an integer is a palindrome by comparing its string representation with its reversed string.
* calculate\_sum\_of\_digits(number): Computes the sum of the digits of an integer by iterating through its string representation.
* determine\_employee\_eligibility(employee\_id): Evaluates an employee ID against the eligibility criteria and returns a list of applicable group memberships.
* process\_employee\_ids(employee\_ids): Implements the divide-and-conquer algorithm to process a list of employee IDs.
* main(input\_file, output\_file): Orchestrates the system, including file I/O, eligibility processing, and report generation.

**4. Implementation Details**

* The system is implemented in Python 3.7+.
* File I/O operations are performed using Python's built-in file handling capabilities.
* Error Handling is implemented for FileNotFoundError and ValueError exceptions.
* Random sampling of eligible IDs is achieved using the random.sample() function.
* Output formatting adheres to the specified requirements.

**5. Complexity Analysis**

* Time Complexity:
  + The process\_employee\_ids() function exhibits a time complexity of O(n log n) due to the divide-and-conquer approach, where n is the number of employee IDs.
  + The is\_number\_palindrome() and calculate\_sum\_of\_digits() functions have a constant time complexity of O(k), where k is the number of digits in the ID (k=5).
  + The creation of the output file has a time complexity of O(n) due to iterating through the eligibility dictionary.
  + Hence total combined time complexity **O(n log n).**
* Space Complexity:
  + The recursive calls in process\_employee\_ids() result in a space complexity of O(log n) due to the call stack.
  + The eligibility\_dict & as a result entire algorithm has a space complexity of **O(n).**

6. **Rationale**

* The divide-and-conquer paradigm was chosen to enhance processing efficiency, particularly for large datasets.
* Functional decomposition promotes code modularity, readability, and maintainability.
* The use of appropriate data structures (dictionaries, lists) facilitates efficient data storage and retrieval.

**7. Testing and Validation**

* Unit Tests: Individual functions are tested to ensure correctness.
* All Ids are validated correcting categorized into groups as required. Ex. All palindrome numbers are moved to palindrome group -1.