1. Research

Problem Definition:

The goal of this project is to develop an application that helps a company assess whether employees are eligible for promotion. The decision is based on two main criteria:

-Years of Experience: A minimum required number of years to qualify for a promotion.

- Performance Rating: Employees need to have a performance rating above or equal to a certain threshold (between 1 and 5).

Once eligibility is confirmed, the application also computes the new salary for the employee based on their performance. This ensures that the promotion is tied not only to experience but also to individual performance, encouraging high standards within the organization.

Key Requirements:

- Input Handling: Gather required experience and performance rating from the company and individual employee data.

- Eligibility Check: Check whether the employee meets the experience and performance criteria.

- Salary Adjustment: Calculate the new salary based on the performance of the employee.

- User Interaction: Allow the system to check multiple employees and store all results for review at the end of the program.

Websites we used were - Quora.com, Stackoverflow.com, w3schools.com, geeksforgeeks, google ai and Github for publishing.

Shortcuts for the code

Shortcut:

Ye – year of experience

Re – required experience

Pr – performance rating

rR – required rating

cS – current salary

nS – new salary

---

2. Analysis

The program handles a straightforward flow for determining employee promotion eligibility:

- The company provides the required experience and performance rating.

- Employees provide their personal experience, rating, and salary details.

- Based on the comparison between employee experience and company criteria, the program identifies promotion eligibility.

- Eligible employees are given a salary increment. The amount of increment depends on the performance rating (higher ratings lead to a larger salary boost).

Important Components:

- Eligibility Check: This is a conditional check comparing employee data with company criteria.

- Salary Calculation: Two separate conditions for salary calculation—major or minor increase depending on the performance rating.

- Iteration: The user can check multiple employees in a single session, and all data is stored in an array for final output.

---

3. Ideate

Several ideas were considered for building and improving this system:

1. Input Validation: Ensure that inputs such as years of experience and performance ratings are valid (i.e., ratings between 1 to 5).

2. Modular Design: Breaking the system into reusable functions for eligibility checks and salary calculations.

3. Extensibility: The system should be able to easily accommodate additional criteria (e.g., education level, skill set) for future enhancement.

4. User Experience: Including color-coded messages (green for success, red for failure, yellow for prompts) makes it easier to communicate with users effectively.

5. Data Storage: The system uses an array to store employee data, but alternative storage mechanisms such as file I/O could be explored for persistence.

---

4. Build

Struct Design:

The `Employee` struct was designed to encapsulate:

- `ye`: Years of experience.

- `pr`: Performance rating (1 to 5 scale).

- `eligibility`: Whether the employee is eligible ('Y' or 'N').

- `cS`: Current salary.

- `nS`: New salary (post-promotion’

Control Flow:

The program uses a `do-while` loop to allow users to repeatedly check employees. The loop ends either when the user chooses to exit or the array limit of 100 employees is reached.

Salary Calculation:

For employees who are eligible, the salary is increased based on performance:

- Performance Rating ≥ 3.0: Major salary increase (`current salary \* 2`).

- Performance Rating < 3.0: Minor salary increase (`current salary \* 1.505`).

---

5. Test

The testing phase focused on verifying different scenarios:

1. Basic Eligibility Check: Employees with sufficient experience and rating should be promoted, while others should not.

2. Salary Computation: Ensured that the salary increment works as expected for high and low performers.

3. Multiple Employee Entries: Verified that the program can correctly handle multiple employees and store results in the array.

4. Edge Cases: Tested boundary cases such as:

- Minimum experience and rating exactly equal to the requirement.

- Maximum employee count to ensure no overflow beyond array size.

5. Exit Condition: The program should exit cleanly when prompted by the user.

Test Case Examples:

- Test 1: Employee has exactly the required years and performance rating.

- Test 2: Employee has fewer years of experience but a high performance rating.

- Test 3: Employee has high experience but lower performance rating.

---

6. Implementation

The final implementation involved creating a clean and user-friendly console interface. Key points:

- User Interaction: Input prompts are clear and color-coded for ease of use.

- Code Efficiency: Using a `struct` for employees simplifies the handling of data and allows for easy extension.

- Salary Logic: The salary increment logic based on performance ensures fairness in promotions, encouraging employees to maintain a high standard.

- Employee Data Storage: The system stores all employee data in an array and displays it at the end, giving a summary of all employees processed.

Final Code Flow:

1. Input: The company enters the minimum required years and performance rating. Then, employees provide their data.

2. Eligibility & Salary Calculation: Based on the input, the program checks if the employee is eligible and calculates the salary accordingly.

3. Review: At the end, all employees' promotion status and salary details are displayed.

publish:

We have published code on GITHUB

7 Maintenance:

In this we will apply 25% maintenance rule that is if maintenance is more than 25% then replace the module, upgrade or rebuild the software.

---