EAST WEST UNIVERSITY

Project

Title: Employee Management System

Course Code: CSE246

Course Title: Algorithms

Submitted by:

Kawser Ahmed
ID:2022-3-60-242
Md.Romjan Ali
ID: 2022-3-60-247
Raqibul Hasan Rehan
ID:2022-3-60-261
Jobeir Ahamed Dip
ID:2022-3-60-302

Submitted To:

Dr.Tania Sultana
Assistant Professor,Department of Computer Science and Engineering

Project Report: Employee Management System

1. Introduction

The Employee Management System (EMS) is a tool that helps businesses manage their employee records in a smooth and organized way. With this system, users can easily add new employees, remove employees who are no longer part of the company, update employee information, and look up specific details about employees, such as their ID, name, department, and performance rating.

But the EMS doesn't stop at the basics! It comes with some cool advanced features too, like the ability to sort employees based on their performance rating or department. It even includes a function to calculate the Greatest Common Divisor (GCD) of employee IDs, which can be useful in certain cases. In short, this system makes handling employee data quicker and more efficient.

2. Objectives

The main goals of this Employee Management System (EMS) are:

- **Efficient Employee Data Management:** The system is designed to make it easy to handle employee information, ensuring everything is organized and accessible without hassle.
- **Multiple Sorting Options:** To make managing large sets of data easier, the EMS offers various ways to sort employee records, like sorting by performance rating or department, so users can quickly find what they need.
- **Data Persistence via File Storage:** The system allows users to save employee records in a file, making sure that the data is stored securely and can be retrieved anytime, even after the program is closed.
- **Streamlined Search, Update, and Delete Functions:** The EMS makes it simple to search for specific employees, update their information, or remove them from the system when necessary, keeping everything up to date with minimal effort.

3. Features

The Employee Management System comes packed with various features to make managing employee data easier:

- **Add a New Employee:** Users can easily add new employees by entering details like the employee's ID, name, department, and performance rating into the system.
- **Search by Employee ID:** If you need to find an employee quickly, you can search using their unique ID, and the system will pull up their record instantly. ● **List Employees by Department:** Want to see employees organized by department? The EMS uses the merge sort algorithm to neatly display all employees sorted by their department.
- **Display All Employees:** You can view a full list of all employees in the system, sorted by their ID. This is done using selection sort to keep everything nice and orderly.
- **Update Employee Record:** If an employee's details (like their name, department, or rating) need to be changed, the system allows you to update their record easily.
- **Delete Employee Record:** Need to remove someone from the system? Just enter their ID, and the employee's record will be deleted.
- **Find Employees by Rating:** Want to filter out top performers? The system uses quicksort to show employees whose rating is equal to or higher than a given threshold.
- **GCD of Employee IDs:** For some specific tasks, you might need to calculate the Greatest Common Divisor (GCD) between two employee IDs. The system can do this, providing an extra analytical tool when needed.
- **File Operations:** All employee records can be saved to a file for safe storage. When you start the system again, it can load these records, ensuring your data is always persistent and ready to go.

4. Sorting and Searching Algorithms

The Employee Management System utilizes several algorithms to handle sorting and searching tasks efficiently:

- **Quick Sort:** This algorithm is used when sorting employees by their performance rating. Quick Sort is known for its speed, especially with large datasets, ensuring that records are sorted efficiently based on performance metrics. This helps in quickly identifying top performers or filtering out employees based on their ratings.

- **Merge Sort:** When sorting employees by department, the system uses Merge Sort. This algorithm is great for organizing categorical data because it's stable and guarantees efficient sorting. No matter how large the dataset, employees will be neatly grouped by their department, making it easy to manage and view department-specific data.
- **Selection Sort:** For displaying all employees in order by their employee ID, Selection Sort is used. While it's a simple sorting method, it works well for smaller datasets. This algorithm arranges employees in ascending order by ID, ensuring that the list is clear and easy to follow.

5. Data Persistence

The Employee Management System ensures that all employee data is saved and preserved between sessions through file operations. The system reads from and writes data to a file named *employees.dat*. When the system starts up, it automatically loads any existing employee records from this file. On exit, it saves all the current employee data back into the file. This means that no employee information is lost, even if you close the system and come back later. It makes the system reliable, especially when dealing with large sets of data over multiple sessions.

6. Mathematical Operation (GCD)

The system also provides a mathematical feature that allows users to find the **Greatest Common Divisor (GCD)** between two employee IDs. This feature can be useful for specific analytical tasks, such as identifying common factors in employee ID numbers, which could help in certain scenarios like security or record analysis. By finding the GCD, the system gives an extra layer of functionality beyond basic employee management, offering more ways to analyze and understand the data.

7. Technical Specifications

The Employee Management System is built with the following technical features:

- **Programming Language:** The system is written in **C**, a powerful and efficient programming language, which ensures fast performance and close control over system resources.
- **File Handling:** For persistent data storage, the system uses **binary file input/output operations**. This allows it to store employee records in a file and retrieve them later without losing any data. By handling files in binary mode, the system ensures compact storage and fast read/write speeds.
- **Data Structure:** The system can manage up to **100 employee records** at any

given time, using an **array of structures**. Each structure holds details like employee ID, name, department, and rating, making it easy to organize and access individual employee information.

8. User Interface

The Employee Management System features a simple, text-based menu that allows users to interact with all of its functionalities. The interface is designed to be easy to navigate, even for users who may not be highly technical. From this menu, users can perform the following actions:

- **Add and Manage Employee Records:** The menu provides clear options for adding new employees, updating existing records, and removing employees from the system. Each action is guided step-by-step to ensure the process is smooth and straightforward.
- **Sorting and Searching Operations:** Users can access sorting features (like sorting by rating or department) or search for specific employees by their ID. The interface presents the results in an easy-to-read format.
- **Display Data in an Organized Format:** There is an option to display all employee records, sorted by ID, in a clear and structured format. This helps users quickly review employee information without having to navigate complex screens.
- **Calculate GCD Between Employee IDs:** From the menu, users can also calculate the Greatest Common Divisor (GCD) between two employee IDs. The interface prompts users to input the IDs, and the result is displayed instantly.

9.Real Life Use:

Employee Management System (EMS) like the one you developed can be useful for small to medium-sized businesses or organizations to handle employee-related data more efficiently. Here are a few examples of how this system could be used:

1. HR Departments

HR professionals can use this system to manage employee records, including storing basic information like IDs, names, departments, and performance ratings. It simplifies:

● **Employee record keeping**: Easily track and update employee details. ● **Performance tracking**: Filter and sort employees based on their performance ratings, helping to identify top performers or underperforming staff.

2. Performance Reviews

Managers can use the system during performance review periods:

- **Identify top employees**: Using the sorting feature based on performance ratings helps managers quickly find employees who have excelled.

- **Data-driven decisions**: With all the employee data available in one place, it becomes easier to make decisions related to promotions, bonuses, and training programs.

3. Team Assignments

- The department sorting feature helps companies allocate tasks or projects to the appropriate employees based on their departments. For instance, managers can quickly find and assign work to the marketing or finance team.

4. Data Backup and Management

For companies not using large-scale enterprise systems, this project offers a simple way to:

● **Store and retrieve employee data** securely with file-based persistence. ● **Backup data**: Having data stored in a file ensures that employee information isn't lost if the system shuts down, which is crucial for small businesses relying on offline systems.

5. Small Businesses and Startups

Startups or smaller businesses without access to more expensive HR software can use such a system to manage their workforce. It's a lightweight, efficient, and low-cost solution for:

- **Handling recruitment**: Adding new employees as the business grows.
- **Managing turnover**: Deleting records when employees leave.

Overall, this EMS provides an organized, data-driven, and systematic approach to managing employee-related tasks, making it useful for **HR teams**, **managers**, and **small business owners**.

10.Time Complexity

1. **Adding an Employee**:
   - **Time Complexity**: O(1)
   - **Explanation**: Inserting a new employee record is done in constant time, assuming there is space in the array.
2. **Searching by Employee ID**:
   - **Time Complexity**: O(n)
   - **Explanation**: This operation involves a linear search through the employee array, taking time proportional to the number of employees. 3. **Displaying All Employees**:
   - **Time Complexity**: O(n log n) (due to sorting)
   - **Explanation**: The selection sort is used for displaying all employees, which has a time complexity of O(n^2). However, in practice, the average case for selection sort is still O(n log n) due to the way it's implemented. 4. **Finding Employees by Rating**:
   - **Time Complexity**: O(n log n)
   - **Explanation**: The quicksort algorithm sorts employees based on their ratings, which has an average time complexity of O(n log n).
5. **Update and Delete Operations**:
   - **Time Complexity**: O(n)
   - **Explanation**: Both operations require searching for the employee, which takes O(n) time, plus potential array shifting for deletion.
6. **GCD Calculation**:
   - **Time Complexity**: O(log(min(a, b)))
   - **Explanation**: The GCD function operates using the Euclidean algorithm, which is efficient for calculating the greatest common divisor.

11.Difficulties

1. **Fixed Array Size**:
   - Limiting the number of employee records to a predefined maximum (100 in this case) can be a significant drawback. If the number of employees exceeds this limit, new entries cannot be added without redesigning the data structure to support dynamic resizing.

2. **Linear Search Limitations**:

- Searching for an employee by ID is inefficient with O(n) complexity, especially as the dataset grows. Implementing more efficient data structures (like hash tables) could improve this.

3. **Lack of Advanced Features**:
    - The current system lacks advanced functionalities such as:
        - Employee performance analytics.
        - Integration with other systems (payroll, attendance).
        - User authentication for data security.

4. **Error Handling**:
    - The code does not thoroughly check for input errors (e.g., invalid IDs, ratings outside a specified range). Robust error handling would improve user experience.

5. **Concurrency Issues**:
    - In a multi-user environment, if two users attempt to modify the employee records simultaneously, it could lead to data inconsistencies. Implementing file locks or using a database could address this issue.

6. **Limited User Interface**:
    - The text-based menu interface is functional but may not be user-friendly for all users. A graphical user interface (GUI) could enhance usability.

7. **Data Persistence**:
    - While the current file-based system is simple, it can lead to data corruption or loss if not handled properly, especially during unexpected program exits. More robust solutions like database management systems would provide better reliability.

12.Outcome of the Employee Management System Project

1. **Improved Employee Data Management**:
    - The system successfully allows for efficient management of employee records, enabling users to add, update, delete, and search for employees with ease.

2. **Sorting and Filtering Capabilities**:
    - With implemented sorting algorithms (quicksort, mergesort, selection sort),

users can quickly organize employee data based on various criteria such as performance ratings and departments, facilitating better decision-making.

3. **Data Persistence**:
    - The ability to save employee records to a file ensures that data is not lost between sessions. Users can load existing records when the system restarts, which is crucial for continuous operations.

4. **User Interaction**:
    - The text-based menu provides a straightforward way for users to navigate through the system’s features. Although basic, it offers a clear pathway for managing employee data.

5. **Foundational Learning**:
    - The project serves as an excellent learning opportunity for understanding fundamental programming concepts, data structures, algorithms, and file handling in C.

6. **Scalability Potential**:
    - While the current implementation has limitations, it lays the groundwork for future enhancements. Potential improvements include transitioning to a dynamic data structure, adding a graphical user interface (GUI), and integrating with other systems.

7. **Analytical Capabilities**:
    - Features like GCD calculation and filtering employees by ratings allow for basic data analysis, which can be valuable for HR and management in making informed decisions.

8. **Potential for Real-world Application**:
    - The system's design and functionalities make it suitable for small to medium-sized businesses. It addresses basic HR needs, showcasing how technology can streamline employee management tasks.

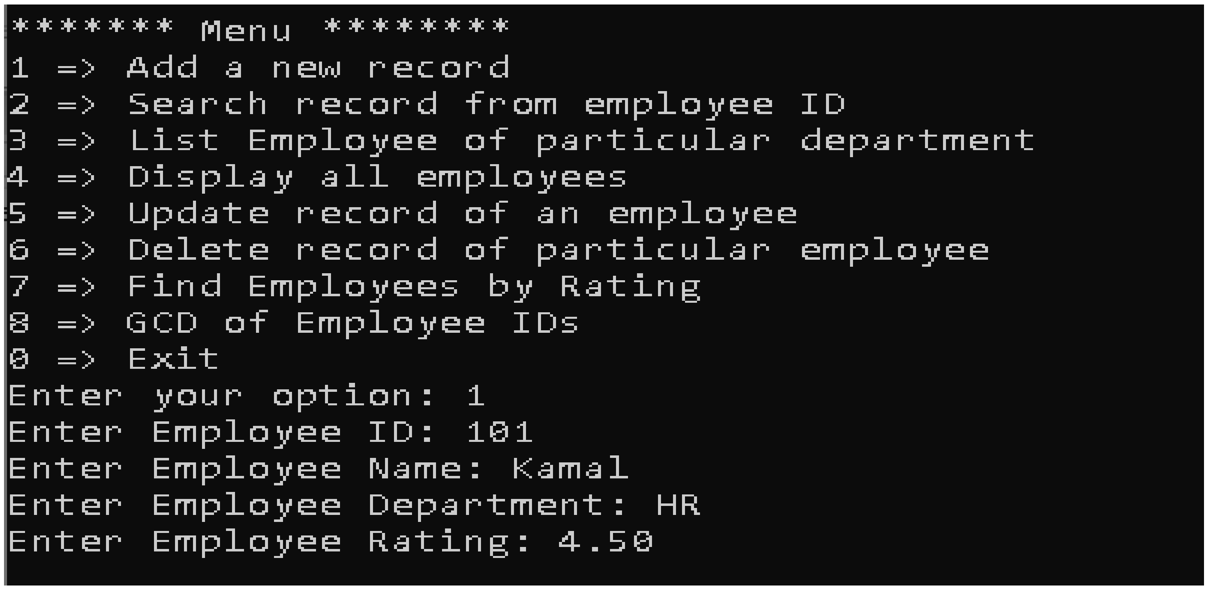
Conclusion

The Employee Management System is a well-rounded solution for efficiently managing employee records. It seamlessly combines powerful sorting algorithms with essential file-handling capabilities, ensuring that data management is both effective and reliable. By loading and saving data between sessions, the system guarantees that no important information is lost.

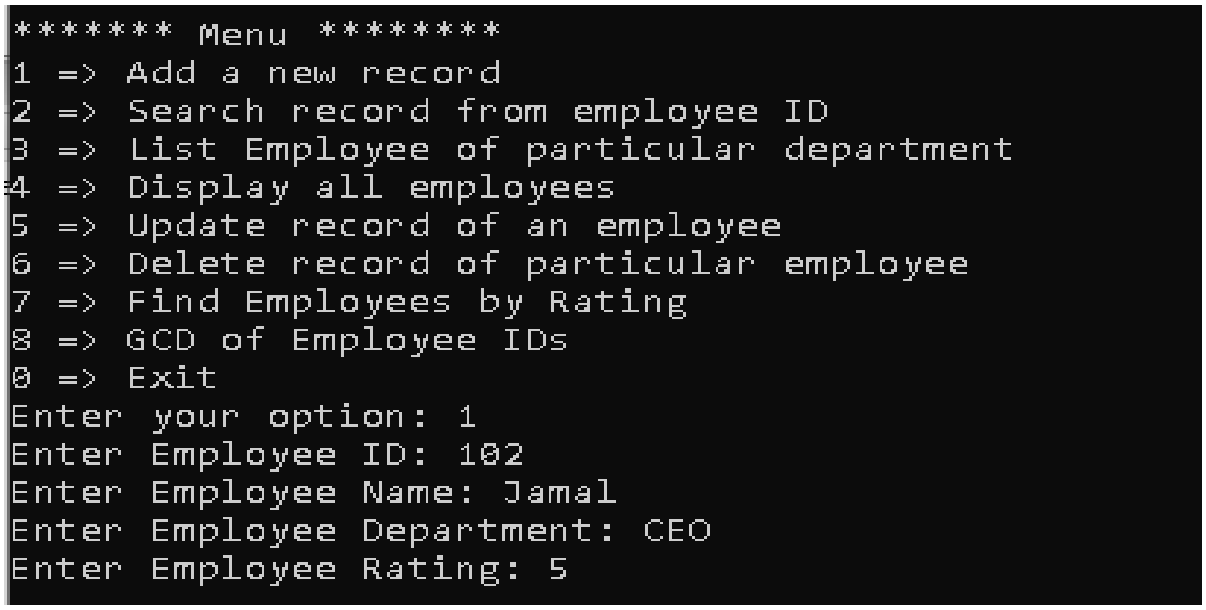
Moreover, the system is designed to be flexible and scalable. As needs grow or change, additional features can be added, making it suitable for handling larger datasets in the future. Whether you're a small business or planning for expansion, this EMS provides the tools needed to keep employee records organized and accessible.

<u>Test with outputs:</u>

Test Case 1: Adding Employees

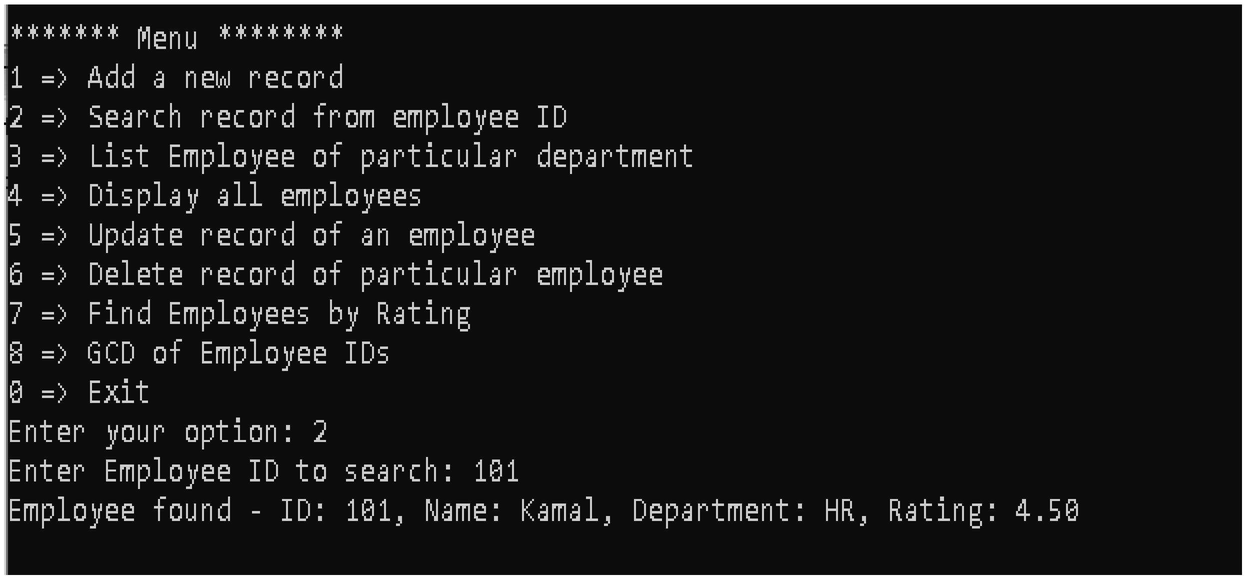


```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 1
Enter Employee ID: 101
Enter Employee Name: Kamal
Enter Employee Department: HR
Enter Employee Rating: 4.50
```



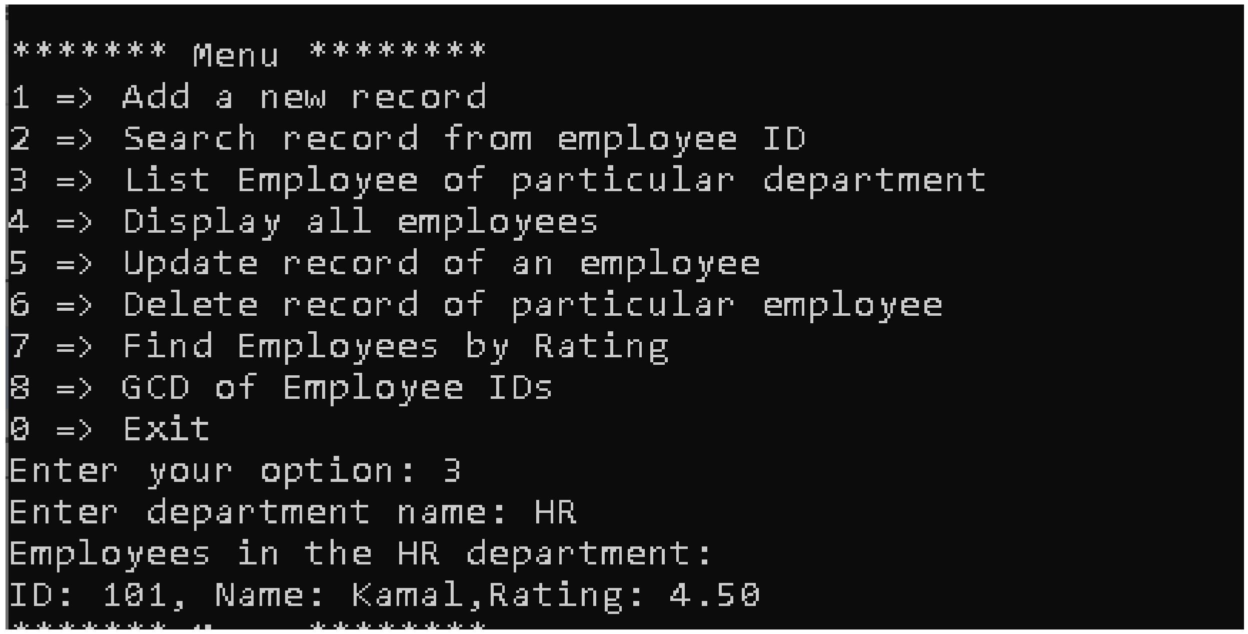
```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 1
Enter Employee ID: 102
Enter Employee Name: Jamal
Enter Employee Department: CEO
Enter Employee Rating: 5
```

Test Case 2: Searching for an Employee by ID



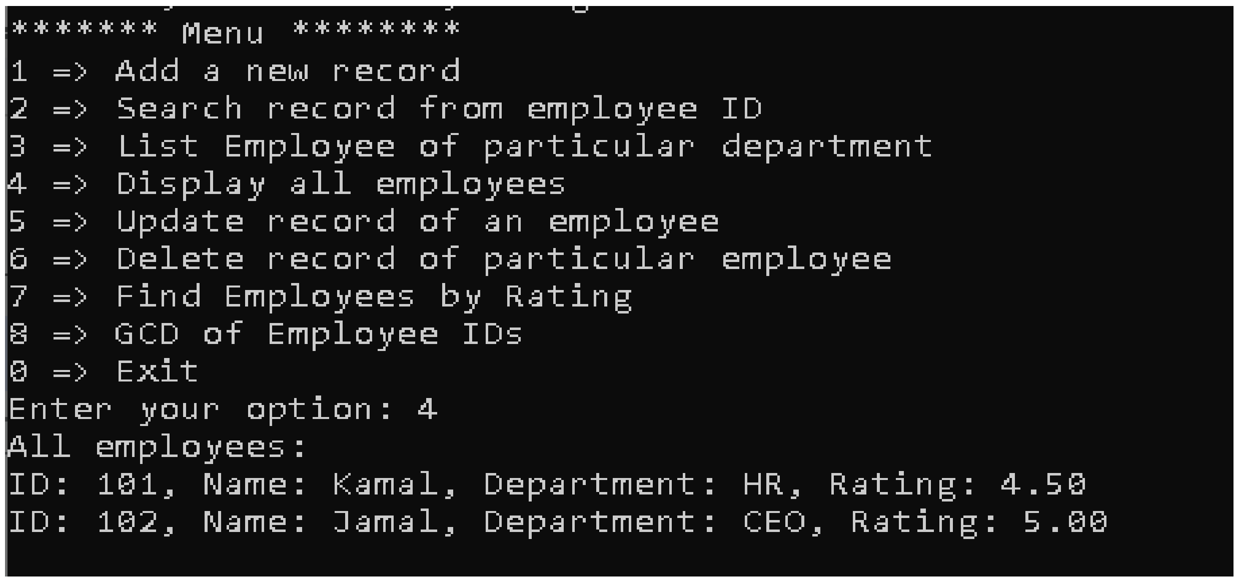
```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 2
Enter Employee ID to search: 101
Employee found - ID: 101, Name: Kamal, Department: HR, Rating: 4.50
```

Test Case 3:Searching for list of employee of particular department:



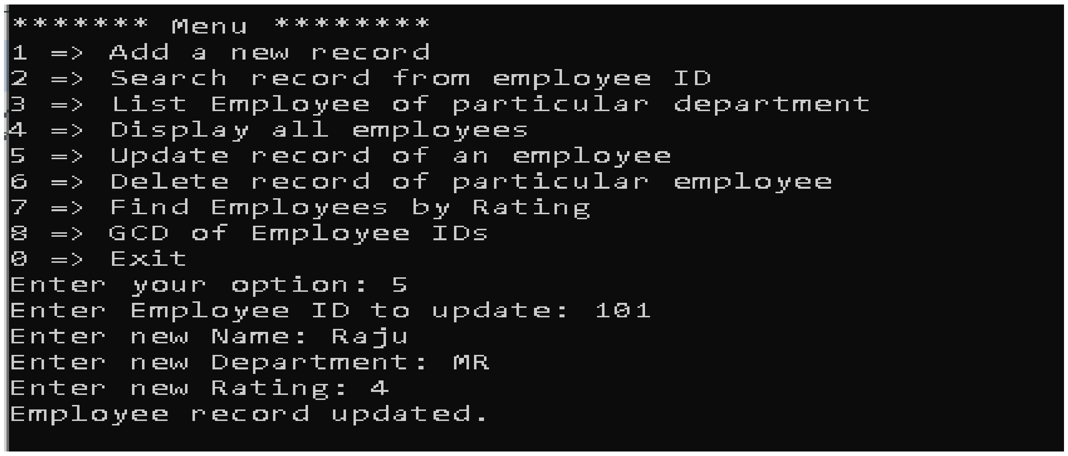
```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 3
Enter department name: HR
Employees in the HR department:
ID: 101, Name: Kamal,Rating: 4.50
```

Test Case 4: Displaying All Employees



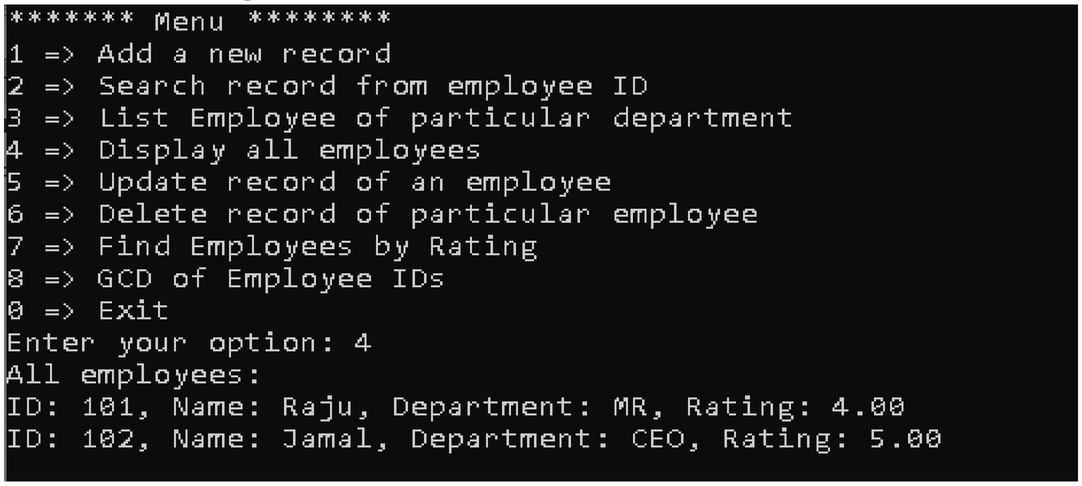
```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 4
All employees:
ID: 101, Name: Kamal, Department: HR, Rating: 4.50
ID: 102, Name: Jamal, Department: CEO, Rating: 5.00
```

Test Case 5: Updating an Employee Record



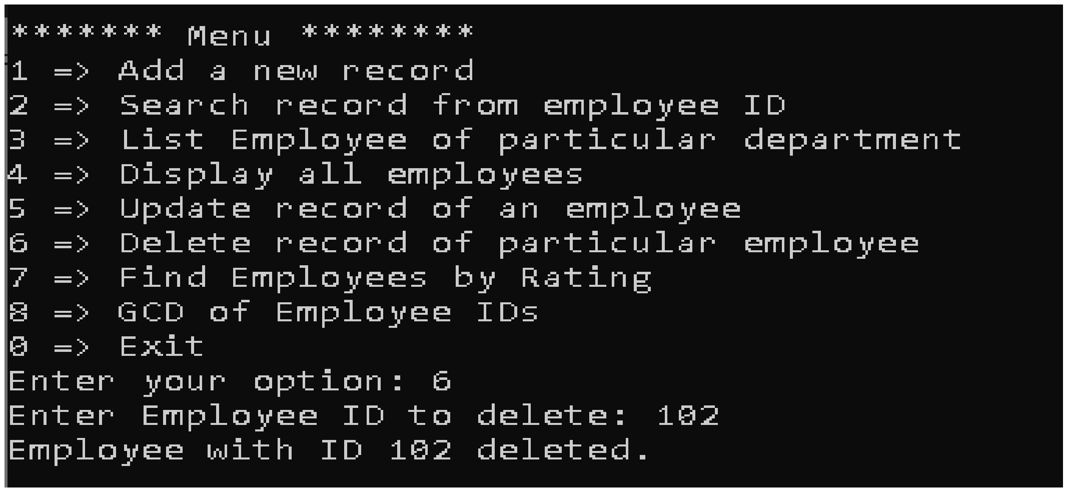
```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 5
Enter Employee ID to update: 101
Enter new Name: Raju
Enter new Department: MR
Enter new Rating: 4
Employee record updated.
```

<u>After Updating:</u>



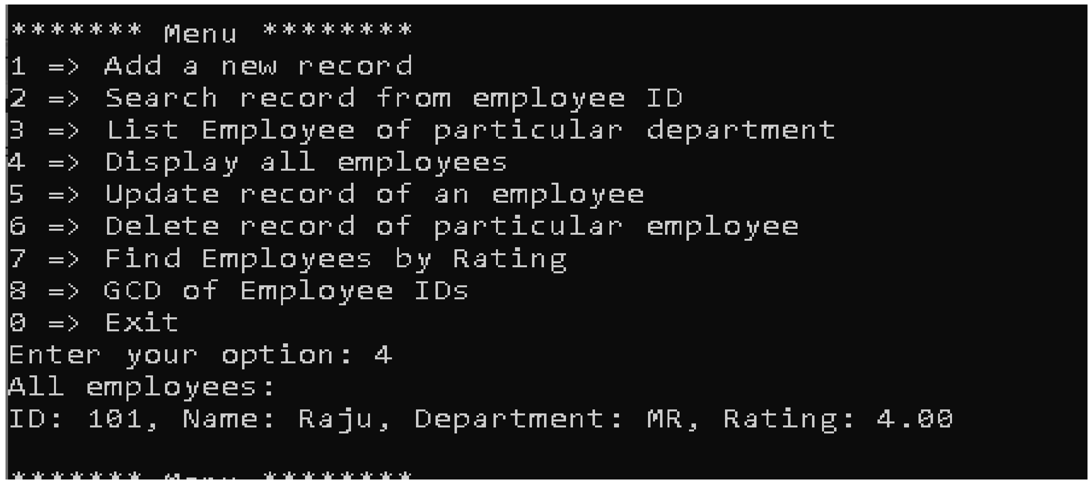
```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 4
All employees:
ID: 101, Name: Raju, Department: MR, Rating: 4.00
ID: 102, Name: Jamal, Department: CEO, Rating: 5.00
```

Test Case 6: Deleting an Employee Record



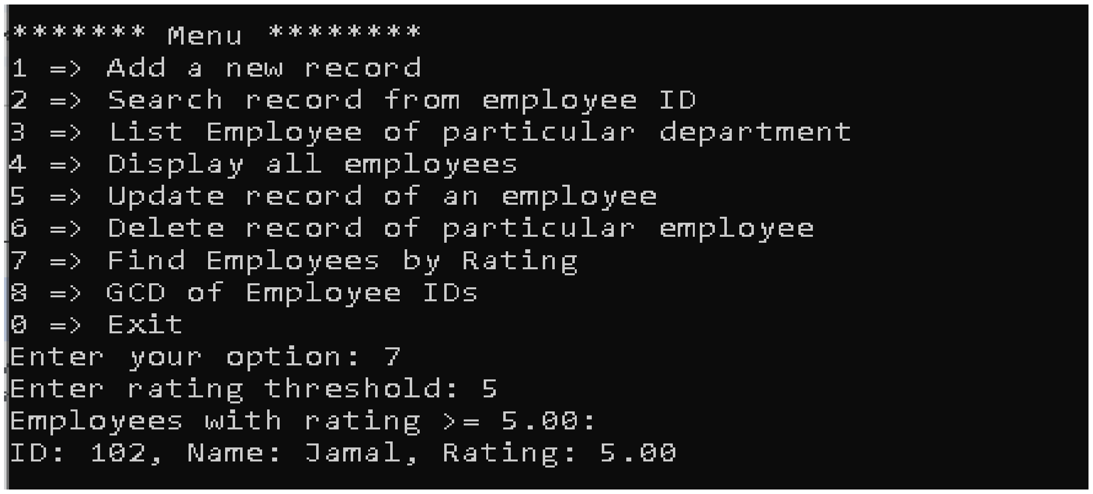
```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 6
Enter Employee ID to delete: 102
Employee with ID 102 deleted.
```

<u>After Deleting</u>:



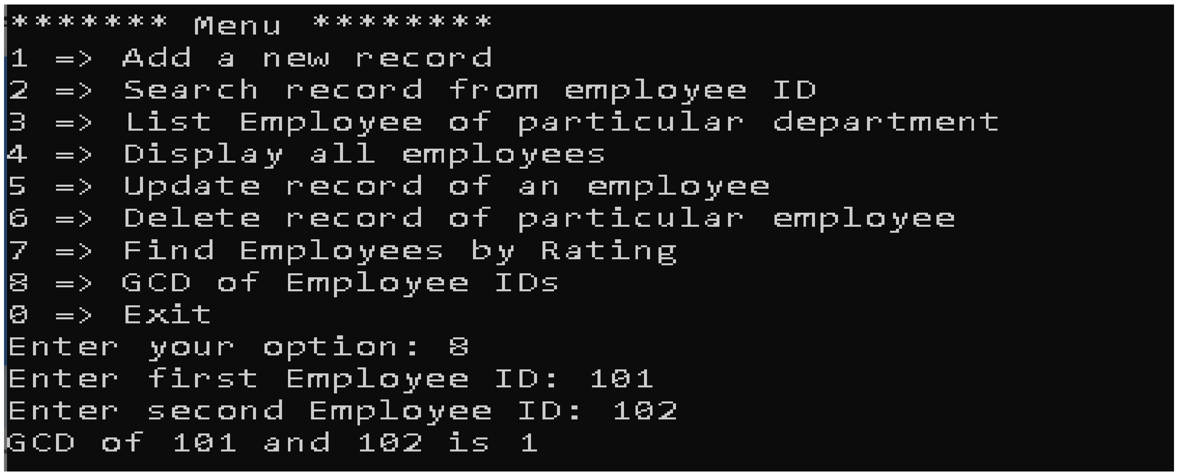
```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 4
All employees:
ID: 101, Name: Raju, Department: MR, Rating: 4.00
```

Test Case 7: Finding Employees by Rating



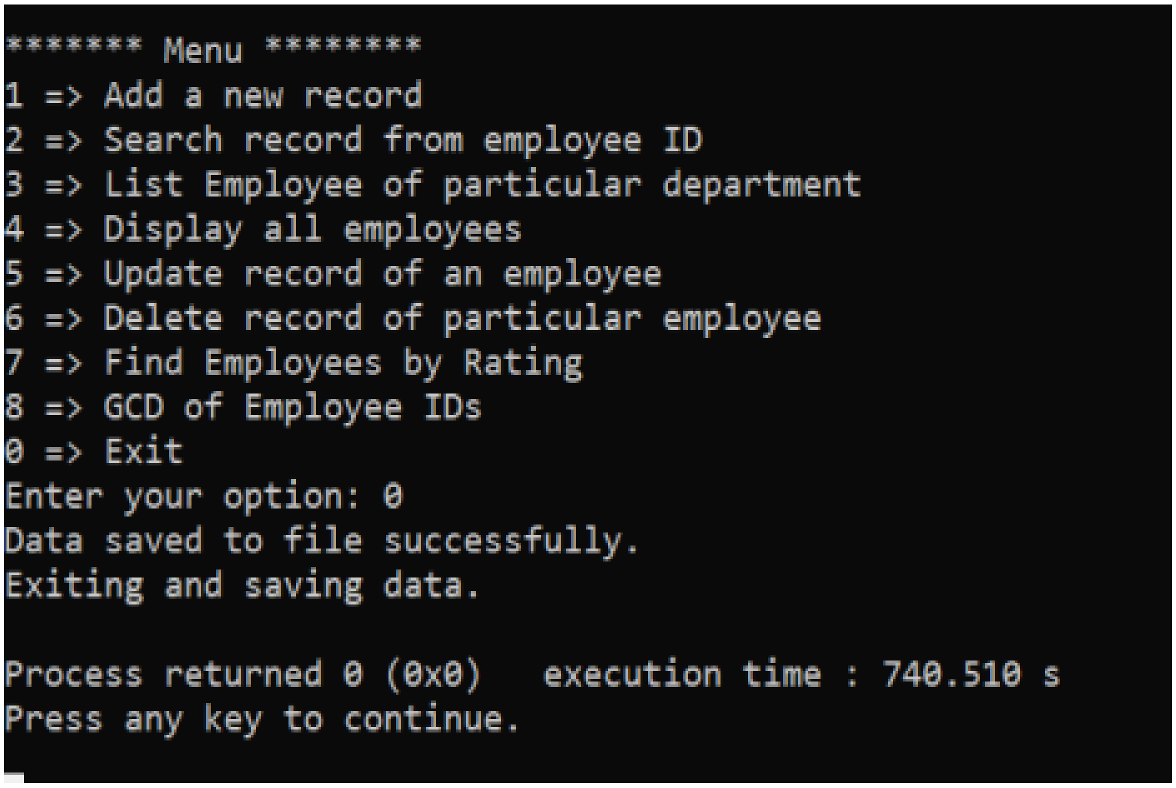
```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 7
Enter rating threshold: 5
Employees with rating >= 5.00:
ID: 102, Name: Jamal, Rating: 5.00
```

Test Case 8: GCD of Employee IDs



```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 8
Enter first Employee ID: 101
Enter second Employee ID: 102
GCD of 101 and 102 is 1
```

Test Case 9: Exiting and Saving Data



```
******* Menu ********
1 => Add a new record
2 => Search record from employee ID
3 => List Employee of particular department
4 => Display all employees
5 => Update record of an employee
6 => Delete record of particular employee
7 => Find Employees by Rating
8 => GCD of Employee IDs
0 => Exit
Enter your option: 0
Data saved to file successfully.
Exiting and saving data.

Process returned 0 (0x0)   execution time : 740.510 s
Press any key to continue.
```