Assignment Report

Course: PROGRAMMING FOR PROBLEM SOLVING (CSS_1001)

Done by: **Yuhanth P** (Registration number: 251580052176)

Title: Case Study on Smart Vehicle Parking & Revenue Management System, Digital Library with Fine & Recommendation, Hostel Mess Billing & Attendance Tracker

Case Study 1: Smart Vehicle Parking & Revenue Management System

Problem Statement

The objective of this project is to design and simulate a Smart Vehicle Parking and Revenue Management System for a shopping mall using the C programming language. The system should use a 2D array to represent the parking layout, where rows indicate floors and columns indicate parking slots.

Each slot can store details of a parked vehicle using a structure containing:

- Vehicle number (string)
- Vehicle type (bike/car/truck/other)
- Entry and exit time

The program should allow multiple vehicles to enter, exit, and calculate parking charges based on vehicle type.
At the end of the day, it should display a summary report showing:

- Total revenue earned
- Percentage of parking slots used

Algorithm / Approach

Step 1: Define Data Structures

- Create a structure named Vehicle with members:

```
struct Vehicle {
    char number[20];
    char type[10];    // "Bike", "Car", "Truck"
    int entryTime;
    int exitTime;
    int isOccupied;   // 1 if slot is filled, 0 if empty
};
```

- Declare a 2D array (e.g., struct Vehicle parking[FLOORS][SLOTS];) to represent multiple floors and slots per floor.

Step 2: Initialize Parking Slots

- Use nested loops to initialize all slots as empty (isOccupied = 0).
- Example: 3 floors with 10 slots each = 30 parking spaces total.

Step 3: Display Menu

- Use a while loop and a menu to repeatedly perform actions until the user exits:
    1. Park a vehicle (Entry)
    2. Exit a vehicle
    3. Display parking status
    4. End of day report
    5. Exit program

Step 4: Vehicle Entry

- Ask for vehicle details (number, type, entry time).
- Search the parking 2D array for the first available (empty) slot.
- If found:
    - Store vehicle details in that slot.
    - Set isOccupied = 1.
    - Print the floor and slot number assigned.
- If no slots are available, display “Parking Full.”

Step 5: Vehicle Exit

- Ask for the vehicle number to identify the slot.
- Use string comparison (strcmp()) to find the vehicle in the parking array.
- Once found:
    - Ask for exit time.
    - Calculate parking duration (exitTime - entryTime).
    - Determine charges based on vehicle type, for example:
        - Bike = ₹10/hour
        - Car = ₹20/hour
        - Truck = ₹30/hour

        - Others = 15/hour
    - Display the total charge.
    - Add the charge to totalRevenue.
    - Mark the slot as empty (isOccupied = 0).

Step 6: Display Parking Status

- Use nested for loops to print the current state of the parking area.
- Show which slots are occupied and which are available on each floor.

Step 7: End-of-Day Report

- Calculate:
    - Total slots: number of floors × number of slots per floor.
    - Used slots: count of all slots that are occupied.
    - Usage percentage: (usedSlots / totalSlots) × 100.
- Display:
    - Total slots and used slots
    - Parking usage percentage
    - Total revenue collected

Step 8: Exit Program

- Exit the menu loop.

Program Code

```c
#include <stdio.h>
#include <string.h>
#define FLOORS 3
#define SLOTS 10

// Structure to store vehicle details
struct Vehicle {
    char number[20];
    char type[10];    // Bike, Car, Truck
    int entryTime;
    int exitTime;
    int isOccupied;   // 1 = slot filled, 0 = empty
};

// Function declarations
void initialize(struct Vehicle parking[FLOORS][SLOTS]);
void parkVehicle(struct Vehicle parking[FLOORS][SLOTS]);
void exitVehicle(struct Vehicle parking[FLOORS][SLOTS], int *totalRevenue);
void displayStatus(struct Vehicle parking[FLOORS][SLOTS]);
void endOfDayReport(struct Vehicle parking[FLOORS][SLOTS], int totalRevenue);

int main() {
    struct Vehicle parking[FLOORS][SLOTS];
    int choice, totalRevenue = 0;
    initialize(parking);

    do {
        printf("\n====== SMART VEHICLE PARKING SYSTEM ======\n");
        printf("1. Park Vehicle (Entry)\n");
        printf("2. Exit Vehicle\n");
        printf("3. Display Parking Status\n");
        printf("4. End of Day Report\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar(); // To clear newline from input buffer

        switch(choice) {
            case 1:
                parkVehicle(parking);
                break;
            case 2:
                exitVehicle(parking, &totalRevenue);
                break;
            case 3:
                displayStatus(parking);
                break;
```

```
            case 4:
                endOfDayReport(parking, totalRevenue);
                break;
            case 5:
                printf("\nThank you for using the Parking System!\n");
                break;
            default:
                printf("\nInvalid choice. Please try again.\n");
        }
    } while(choice != 5);

    return 0;
}

// Initialize all parking slots as empty
void initialize(struct Vehicle parking[FLOORS][SLOTS]) {
    for(int i = 0; i < FLOORS; i++) {
        for(int j = 0; j < SLOTS; j++) {
            parking[i][j].isOccupied = 0;
        }
    }
}

// Function to park a vehicle
void parkVehicle(struct Vehicle parking[FLOORS][SLOTS]) {
    char number[20], type[10];
    int entry, parked = 0;

    printf("\nEnter Vehicle Number: ");
    fgets(number, sizeof(number), stdin);
    number[strcspn(number, "\n")] = '\0';

    printf("Enter Vehicle Type (bike/car/truck): ");
    fgets(type, sizeof(type), stdin);
    type[strcspn(type, "\n")] = '\0';

    printf("Enter Entry Time (in hours, e.g., 10 for 10AM): ");
    scanf("%d", &entry);
    getchar();  // To clear newline from input buffer

    // Find first empty slot
    for(int i = 0; i < FLOORS && !parked; i++) {
        for(int j = 0; j < SLOTS && !parked; j++) {
            if(parking[i][j].isOccupied == 0) {
                strcpy(parking[i][j].number, number);
                strcpy(parking[i][j].type, type);
                parking[i][j].entryTime = entry;
                parking[i][j].isOccupied = 1;
```

```
                parked = 1;
                printf("\nVehicle parked at Floor %d, Slot %d\n", i + 1, j +
1);
            }
        }
    }

    if(!parked)
        printf("\nParking Full! No available slots.\n");
}

// Function to exit a vehicle and calculate charge
void exitVehicle(struct Vehicle parking[FLOORS][SLOTS], int *totalRevenue) {
    char number[20];
    int exitTime, duration, charge = 0, found = 0;

    printf("\nEnter Vehicle Number to Exit: ");
    fgets(number, sizeof(number), stdin);
    number[strcspn(number, "\n")] = '\0';

    for(int i = 0; i < FLOORS; i++) {
        for(int j = 0; j < SLOTS; j++) {
            if(parking[i][j].isOccupied == 1 && strcmp(parking[i][j].number,
number) == 0) {
                found = 1;
                printf("Enter Exit Time (in hours): ");
                scanf("%d", &exitTime);
                getchar();  // To clear newline from input buffer

                duration = exitTime - parking[i][j].entryTime;
                if(duration <= 0)
                    duration = 1; // minimum 1 hour charge

                // Calculate charge based on vehicle type
                if(strcmp(parking[i][j].type, "bike") == 0)
                    charge = duration * 10; // 10 rupees per hour for bike
                else if(strcmp(parking[i][j].type, "car") == 0)
                    charge = duration * 20; // 20 rupees per hour for car
                else if(strcmp(parking[i][j].type, "truck") == 0)
                    charge = duration * 30; // 30 rupees per hour for truck
                else
                    charge = duration * 15; // default (15 rupees per hour if
neither)

                printf("\nVehicle Found at Floor %d, Slot %d\n", i + 1, j +
1);
                printf("Parking Duration: %d hour(s)\n", duration);
                printf("Total Charge: Rupees %d\n", charge);
```

```
                *totalRevenue += charge;
                parking[i][j].isOccupied = 0; // Slot is now free
                break;
            }
        }
    }

    if(!found)
        printf("\nVehicle not found in the parking lot.\n");
}

// Display current parking status
void displayStatus(struct Vehicle parking[FLOORS][SLOTS]) {
    printf("\n------ Parking Status ------\n");
    for(int i = 0; i < FLOORS; i++) {
        printf("\nFloor %d: ", i + 1);
        for(int j = 0; j < SLOTS; j++) {
            if(parking[i][j].isOccupied)
                printf("[X] ");  // Occupied
            else
                printf("[ ] ");  // Empty
        }
        printf("\n");
    }
}

// Generate End of Day Report
void endOfDayReport(struct Vehicle parking[FLOORS][SLOTS], int totalRevenue) {
    int totalSlots = FLOORS * SLOTS;
    int usedSlots = 0;

    for(int i = 0; i < FLOORS; i++) {
        for(int j = 0; j < SLOTS; j++) {
            if(parking[i][j].isOccupied == 1)
                usedSlots++;
        }
    }

    float usagePercentage = ((float)usedSlots / totalSlots) * 100;

    printf("\n====== END OF DAY REPORT ======\n");
    printf("Total Parking Slots : %d\n", totalSlots);
    printf("Slots Occupied      : %d\n", usedSlots);
    printf("Usage Percentage    : %.2f%%\n", usagePercentage);
    printf("Total Revenue Earned: Rupees %d\n", totalRevenue);
    printf("================================\n");
}
```

Sample Input / Output

====== SMART VEHICLE PARKING SYSTEM ======

1. Park Vehicle (Entry)

2. Exit Vehicle

3. Display Parking Status

4. End of Day Report

5. Exit

Enter your choice: 1

Enter Vehicle Number: TN01AB0001

Enter Vehicle Type (bike/car/truck): bike

Enter Entry Time (in hours, e.g., 10 for 10AM): 10

Vehicle parked at Floor 1, Slot 1

====== SMART VEHICLE PARKING SYSTEM ======

1. Park Vehicle (Entry)

2. Exit Vehicle

3. Display Parking Status

4. End of Day Report

5. Exit

Enter your choice: 1

Enter Vehicle Number: TN02AB0002

Enter Vehicle Type (bike/car/truck): car

Enter Entry Time (in hours, e.g., 10 for 10AM): 12

Vehicle parked at Floor 1, Slot 2

====== SMART VEHICLE PARKING SYSTEM ======

1. Park Vehicle (Entry)

2. Exit Vehicle

3. Display Parking Status

4. End of Day Report

5. Exit

Enter your choice: 1

Enter Vehicle Number: TN03XY0003

Enter Vehicle Type (bike/car/truck): truck

Enter Entry Time (in hours, e.g., 10 for 10AM): 4

Vehicle parked at Floor 1, Slot 3

====== SMART VEHICLE PARKING SYSTEM ======

1. Park Vehicle (Entry)

2. Exit Vehicle

3. Display Parking Status

4. End of Day Report

5. Exit

Enter your choice: 2

Enter Vehicle Number to Exit: TN02AB000

Vehicle not found in the parking lot.

====== SMART VEHICLE PARKING SYSTEM ======

1. Park Vehicle (Entry)

2. Exit Vehicle

3. Display Parking Status

4. End of Day Report

5. Exit

Enter your choice: 2

Enter Vehicle Number to Exit: TN02AB0002

Enter Exit Time (in hours): 19

Vehicle Found at Floor 1, Slot 2

Parking Duration: 7 hour(s)

Total Charge: Rupees 140

====== SMART VEHICLE PARKING SYSTEM ======

1. Park Vehicle (Entry)

2. Exit Vehicle

3. Display Parking Status

4. End of Day Report

5. Exit

Enter your choice: 3

------ Parking Status ------

Floor 1: [X] [ ] [X] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

Floor 2: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

Floor 3: [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

====== SMART VEHICLE PARKING SYSTEM ======

1. Park Vehicle (Entry)

2. Exit Vehicle

3. Display Parking Status

4. End of Day Report

5. Exit

Enter your choice: 4

====== END OF DAY REPORT ======

Total Parking Slots : 30

Slots Occupied : 2

Usage Percentage : 6.67%

Total Revenue Earned: Rupees 140

==========

====== SMART VEHICLE PARKING SYSTEM ======

1. Park Vehicle (Entry)

2. Exit Vehicle

3. Display Parking Status

4. End of Day Report

5. Exit

Enter your choice: 5

Thank you for using the Parking System!

Process returned 0

Conclusion

By developing the Smart Vehicle Parking and Revenue Management System, I learned how to combine several C programming concepts to create a practical application.
I understood how 2D arrays can model real-world grids (like parking floors and slots), and how structures store multiple attributes of each vehicle.
The project helped me practice string operations for vehicle identification, loops for continuous user interaction, and conditional statements for decision-making and billing logic.
Finally, generating a report using calculations like slot usage and total revenue showed me how programming can be applied to data analysis and management systems in real scenarios.

Case Study 2: Digital Library with Fine & Recommendation

Problem Statement

The goal of this project is to design and implement a SIMPLE Digital Library Management System using the C programming language.
The system should store details of books such as title, author, genre, and number of copies. Students should be able to search, issue, and return books. If a book is returned late, the program should calculate a fine. The system should also recommend books to students based on their preferred genre.

Algorithm / Approach

Step 1: Define Data Structure

- Create a structure named Book with members: title, author, genre, and copies.
- Declare an array of Book to store multiple book records in the library.

Step 2: Initialize Data

- Preload the array with sample books to simulate a small library database.
- Each book entry includes its title, author, genre, and the number of available copies.

Step 3: Display Menu

- Use a loop to repeatedly show the main menu until the user chooses to exit.
- Menu options include:
    1. Display all books
    2. Search book
    3. Issue book
    4. Return book
    5. Recommend book
    6. Exit

Step 4: Implement Search Function

- Accept a keyword (title or author) from the user.
- Use string comparison functions (strcmp() or strstr()) to find matching books.
- Display details of any books that match the search term.

Step 5: Issue Book

- Ask the user for the book title.
- If the book exists and copies are available, reduce the number of copies by one.
- Otherwise, display a “Not available” message.

Step 6: Return Book

- Ask for the title of the book being returned.
- Increase the number of copies by one.
- Ask how many days late the return is.
- Use a conditional statement to calculate the fine (e.g., ₹2 per late day).

Step 7: Recommend Books

- Ask for the user’s preferred genre.
- Compare it with the genre field of each book using strcmp().
- Display a list of books that match the given genre.

Step 8: Exit the Program

- Exit the menu loop.

Program Code

```c
#include <stdio.h>
#include <string.h>

// Structure to store book details
struct Book {
    char title[50];
    char author[50];
    char genre[30];
    int copies;
};

// Function declarations
void displayBooks(struct Book library[], int count);
void searchBook(struct Book library[], int count);
void issueBook(struct Book library[], int count);
void returnBook(struct Book library[], int count);
void recommendBook(struct Book library[], int count);

int main() {
    struct Book library[50];
    int count = 5; // Number of books in library
    int choice;

    // Initialize books
    strcpy(library[0].title, "C Programming");
    strcpy(library[0].author, "Dennis Ritchie");
    strcpy(library[0].genre, "Programming");
    library[0].copies = 3;

    strcpy(library[1].title, "The Alchemist");
    strcpy(library[1].author, "Paulo Coelho");
    strcpy(library[1].genre, "Fiction");
    library[1].copies = 2;

    strcpy(library[2].title, "Data Structures");
    strcpy(library[2].author, "Mark Weiss");
    strcpy(library[2].genre, "Education");
    library[2].copies = 4;

    strcpy(library[3].title, "Harry Potter");
    strcpy(library[3].author, "J.K. Rowling");
    strcpy(library[3].genre, "Fantasy");
    library[3].copies = 5;

    strcpy(library[4].title, "Rich Dad Poor Dad");
    strcpy(library[4].author, "Robert Kiyosaki");
    strcpy(library[4].genre, "Finance");
```

```c
    library[4].copies = 3;

    // Menu-driven program
    do {
        printf("\n====== DIGITAL LIBRARY SYSTEM ======\n");
        printf("1. Display All Books\n");
        printf("2. Search Book\n");
        printf("3. Issue Book\n");
        printf("4. Return Book\n");
        printf("5. Recommend Book\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar(); // To clear newline from input buffer

        switch(choice) {
            case 1:
                displayBooks(library, count);
                break;
            case 2:
                searchBook(library, count);
                break;
            case 3:
                issueBook(library, count);
                break;
            case 4:
                returnBook(library, count);
                break;
            case 5:
                recommendBook(library, count);
                break;
            case 6:
                printf("\nThank you for using the Digital Library!\n");
                break;
            default:
                printf("\nInvalid choice. Please try again.\n");
        }

    } while(choice != 6);

    return 0;
}

// Function to display all books
void displayBooks(struct Book library[], int count) {
    printf("\nAvailable Books:\n");
    printf("----------------------------------------\n");
    for(int i = 0; i < count; i++) {
```

```
        printf("Title: %s\nAuthor: %s\nGenre: %s\nCopies: %d\n\n",
               library[i].title, library[i].author, library[i].genre,
library[i].copies);
    }
}

// Function to search for a book
void searchBook(struct Book library[], int count) {
    char keyword[50];
    int found = 0;

    printf("\nEnter book title or author to search: ");
    fgets(keyword, sizeof(keyword), stdin);
    keyword[strcspn(keyword, "\n")] = '\0'; // remove newline

    for(int i = 0; i < count; i++) {
        if(strstr(library[i].title, keyword) != NULL ||
strstr(library[i].author, keyword) != NULL) {
            printf("\nBook Found:\nTitle: %s\nAuthor: %s\nGenre: %s\nCopies:
%d\n",
                   library[i].title, library[i].author, library[i].genre,
library[i].copies);
            found = 1;
        }
    }
    if(!found)
        printf("\nNo matching book found.\n");
}

// Function to issue a book
void issueBook(struct Book library[], int count) {
    char title[50];
    int found = 0;

    printf("\nEnter book title to issue: ");
    fgets(title, sizeof(title), stdin);
    title[strcspn(title, "\n")] = '\0';

    for(int i = 0; i < count; i++) {
        if(strcmp(library[i].title, title) == 0) {
            found = 1;
            if(library[i].copies > 0) {
                library[i].copies--;
                printf("Book issued successfully!\n");
            } else {
                printf("Sorry, no copies available.\n");
            }
        }
```

```
    }
    if(!found)
        printf("Book not found in library.\n");
}

// Function to return a book and calculate fine
void returnBook(struct Book library[], int count) {
    char title[50];
    int daysLate, fine;
    int found = 0;

    printf("\nEnter book title to return: ");
    fgets(title, sizeof(title), stdin);
    title[strcspn(title, "\n")] = '\0';

    for(int i = 0; i < count; i++) {
        if(strcmp(library[i].title, title) == 0) {
            found = 1;
            library[i].copies++;
            printf("Enter number of days late (0 if on time): ");
            scanf("%d", &daysLate);
            getchar();  // To clear newline from input buffer
            if(daysLate > 0) {
                fine = daysLate * 2;
                printf("Book returned late. Fine = Rupees %d\n", fine);
            } else {
                printf("Book returned on time. No fine.\n");
            }
        }
    }
    if(!found)
        printf("Book not found in library.\n");
}

// Function to recommend books by genre
void recommendBook(struct Book library[], int count) {
    char genre[30];
    int found = 0;

    printf("\nEnter preferred genre: ");
    fgets(genre, sizeof(genre), stdin);
    genre[strcspn(genre, "\n")] = '\0';

    printf("\nRecommended Books in '%s':\n", genre);
    for(int i = 0; i < count; i++) {
        if(strcmp(library[i].genre, genre) == 0) {
            printf(" - %s by %s\n", library[i].title, library[i].author);
            found = 1;
```

```
        }
    }
    if(!found)
        printf("No recommendations available for this genre.\n");
}
```

Sample Input / Output

====== DIGITAL LIBRARY SYSTEM ======

1. Display All Books

2. Search Book

3. Issue Book

4. Return Book

5. Recommend Book

6. Exit

Enter your choice: 1

Available Books:

Title: C Programming

Author: Dennis Ritchie

Genre: Programming

Copies: 3

Title: The Alchemist

Author: Paulo Coelho

Genre: Fiction

Copies: 2

Title: Data Structures

Author: Mark Weiss

Genre: Education

Copies: 4

Title: Harry Potter

Author: J.K. Rowling

Genre: Fantasy

Copies: 5

Title: Rich Dad Poor Dad

Author: Robert Kiyosaki

Genre: Finance

Copies: 3

====== DIGITAL LIBRARY SYSTEM ======

1. Display All Books

2. Search Book

3. Issue Book

4. Return Book

5. Recommend Book

6. Exit

Enter your choice: 2

Enter book title or author to search: Harry

Book Found:

Title: Harry Potter

Author: J.K. Rowling

Genre: Fantasy

Copies: 5

====== DIGITAL LIBRARY SYSTEM ======

1. Display All Books

2. Search Book

3. Issue Book

4. Return Book

5. Recommend Book

6. Exit

Enter your choice: 3

Enter book title to issue: The Alchemist

Book issued successfully!

====== DIGITAL LIBRARY SYSTEM ======

1. Display All Books

2. Search Book

3. Issue Book

4. Return Book

5. Recommend Book

6. Exit

Enter your choice: 3

Enter book title to issue: The Alchemist

Book issued successfully!

====== DIGITAL LIBRARY SYSTEM ======

1. Display All Books

2. Search Book

3. Issue Book

4. Return Book

5. Recommend Book

6. Exit

Enter your choice: 3

Enter book title to issue: The Alchemist

Sorry, no copies available.

====== DIGITAL LIBRARY SYSTEM ======

1. Display All Books

2. Search Book

3. Issue Book

4. Return Book

5. Recommend Book

6. Exit

Enter your choice: 4

Enter book title to return: The Alchemist

Enter number of days late (0 if on time): 0

Book returned on time. No fine.

====== DIGITAL LIBRARY SYSTEM ======

1. Display All Books

2. Search Book

3. Issue Book

4. Return Book

5. Recommend Book

6. Exit

Enter your choice: 4

Enter book title to return: The Alchemist

Enter number of days late (0 if on time): 10

Book returned late. Fine = Rupees 20

====== DIGITAL LIBRARY SYSTEM ======

1. Display All Books

2. Search Book

3. Issue Book

4. Return Book

5. Recommend Book

6. Exit

Enter your choice: 5

Enter preferred genre: Fantasy

Recommended Books in 'Fantasy':

- Harry Potter by J.K. Rowling

====== DIGITAL LIBRARY SYSTEM ======

1. Display All Books

2. Search Book

3. Issue Book

4. Return Book

5. Recommend Book

6. Exit

Enter your choice: 6

Thank you for using the Digital Library!

Process returned 0

Conclusion

By developing this Digital Library Management System, I learned how to integrate multiple core C programming concepts into a single program. I understood how structures can group related information, how arrays manage collections of data, and how string functions allow searching and comparison.
This project helped to understand how basic data handling and control flow are implemented in real-world applications.

Case Study 3: Hostel Mess Billing & Attendance Tracker

Problem Statement

The objective of this project is to design and implement a Hostel Mess Billing and Attendance Tracker using the C programming language. The system manages student details and their daily mess attendance to calculate monthly mess bills. Each student's details, such as ID, name, and attendance record, are stored using a structure. The program calculates each student's monthly bill based on the number of days they attended the mess. Students who were on leave for more than 5 days receive a discount, and the system can also identify students with unpaid bills at the end of the month.

Algorithm / Approach

Step 1: Define a Structure

Create a structure named Student with members:

```
struct Student {
   int id;
   char name[50];
   int attendance[31];  // 1 = present, 0 = absent
   int totalDays;
   int billPaid;
};
```

This structure stores the student's details, daily attendance, and payment status.

Step 2: Initialize Data

- Create an array of students (e.g., struct Student s[50];).
- Input student details (ID and name).
- Initialize attendance for all days as 0 (absent).

Step 3: Update Attendance

- Use a loop to update daily attendance for all students.
- For each student, record whether they attended the mess (1) or not (0).
- Repeat this for all days in the month.

Step 4: Calculate Total Attendance

- Use a loop to count total attendance days for each student:
- totalDays = sum of attendance[i][day];

Step 5: Calculate Mess Bill

- Assume a fixed rate per day, e.g., ₹100/day.
- Calculate monthly bill:
- bill = totalDays × ratePerDay
- If student was absent for more than 5 days, apply a 10% discount:
- if (30 - totalDays > 5)
- bill = bill - (bill * 0.10);

Step 6: Payment Update

- Ask whether the student has paid the bill (1 = Paid, 0 = Not Paid).
- Store this in billPaid.

Step 7: Identify Students with Unpaid Bills

- Loop through all students and display names and IDs of those whose billPaid == 0.

Step 8: Display Summary

- Print each student's:
    - ID, Name
    - Total days attended
    - Calculated bill
    - Discount (if any)
    - Payment status

Program Code

```c
#include <stdio.h>
#include <string.h>

#define DAYS 7       // Total days in a month (7 for simulation)
#define RATE 100      // Cost per day (Rupees 100)
#define MAX_STUDENTS 20

// Structure to store student details
struct Student {
    int id;
    char name[50];
    int attendance[DAYS];  // 1 = present, 0 = absent
    int totalDays;
    float billAmount;
    int billPaid;          // 1 = paid, 0 = unpaid
};

// Function declarations
void markAttendance(struct Student students[], int n);
void calculateBills(struct Student students[], int n);
void displaySummary(struct Student students[], int n);
void unpaidStudents(struct Student students[], int n);

int main() {
    struct Student students[MAX_STUDENTS];
    int n, choice;

    printf("Enter number of students: ");
    scanf("%d", &n);
    getchar(); // To clear newline from input buffer

    // Input student details
    for(int i = 0; i < n; i++) {
        students[i].totalDays = 0;
        students[i].billPaid = 0;
        printf("\nEnter details for student %d\n", i + 1);
        printf("ID: ");
        scanf("%d", &students[i].id);
        getchar();

        printf("Name: ");
        fgets(students[i].name, sizeof(students[i].name), stdin);
        students[i].name[strcspn(students[i].name, "\n")] = '\0'; // remove
newline

        // Initialize attendance to 0
        for(int d = 0; d < DAYS; d++)
```

```
            students[i].attendance[d] = 0;
    }

    // Menu-driven program
    do {
        printf("\n====== HOSTEL MESS MANAGEMENT SYSTEM ======\n");
        printf("1. Mark Attendance\n");
        printf("2. Calculate Bills\n");
        printf("3. Display Summary\n");
        printf("4. Show Students with Unpaid Bills\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();

        switch(choice) {
            case 1:
                markAttendance(students, n);
                break;
            case 2:
                calculateBills(students, n);
                break;
            case 3:
                displaySummary(students, n);
                break;
            case 4:
                unpaidStudents(students, n);
                break;
            case 5:
                printf("\nExiting system. Thank you!\n");
                break;
            default:
                printf("\nInvalid choice! Try again.\n");
        }
    } while(choice != 5);

    return 0;
}

// Function to mark attendance
void markAttendance(struct Student students[], int n) {
    int day;
    printf("\nEnter the day number (1-%d): ", DAYS);
    scanf("%d", &day);
    getchar();

    if(day < 1 || day > DAYS) {
        printf("Invalid day number.\n");
```

```
        return;
    }

    for(int i = 0; i < n; i++) {
        int present;
        printf("Is %s (ID: %d) present today? (1 = Yes, 0 = No): ",
students[i].name, students[i].id);
        scanf("%d", &present);
        getchar();

        students[i].attendance[day - 1] = present;
    }

    printf("\nAttendance for Day %d recorded successfully!\n", day);
}

// Function to calculate bills
void calculateBills(struct Student students[], int n) {
    for(int i = 0; i < n; i++) {
        int presentDays = 0;

        for(int d = 0; d < DAYS; d++)
            if(students[i].attendance[d] == 1)
                presentDays++;

        students[i].totalDays = presentDays;
        float bill = presentDays * RATE;
        int absentDays = DAYS - presentDays;

        // Discount if absent for more than 5 days
        if(absentDays > 5) {
            float discount = bill * 0.10;
            bill -= discount;
            printf("\n%s (ID: %d) gets a discount of Rupees %.2f for %d days
leave.\n",
                   students[i].name, students[i].id, discount, absentDays);
        }

        students[i].billAmount = bill;

        printf("\n%s (ID: %d): Total Days = %d, Bill = Rupees %.2f\n",
               students[i].name, students[i].id, presentDays, bill);

        printf("Has the student paid the bill? (1 = Yes, 0 = No): ");
        scanf("%d", &students[i].billPaid);
        getchar();
    }
    printf("\nBills calculated and payment status updated!\n");
```

```c
}

// Function to display summary of all students
void displaySummary(struct Student students[], int n) {
    printf("\n====== MONTHLY SUMMARY ======\n");
    printf("ID\tName\t\tDays\tBill(Rupees)\tStatus\n");
    printf("-----------------------------------------------\n");

    for(int i = 0; i < n; i++) {
        printf("%d\t%-12s\t%d\t%.2f\t%s\n",
               students[i].id,
               students[i].name,
               students[i].totalDays,
               students[i].billAmount,
               students[i].billPaid ? "Paid" : "Unpaid");
    }
}

// Function to display unpaid students
void unpaidStudents(struct Student students[], int n) {
    printf("\n====== UNPAID STUDENTS ======\n");
    int found = 0;
    for(int i = 0; i < n; i++) {
        if(students[i].billPaid == 0) {
            printf("ID: %d | Name: %s | Due: Rupees %.2f\n",
                   students[i].id,
                   students[i].name,
                   students[i].billAmount);
            found = 1;
        }
    }
    if(!found)
        printf("All students have paid their bills!\n");
}
```

Sample Input / Output

Enter number of students: 2

Enter details for student 1

ID: 001

Name: Student_1

Enter details for student 2

ID: 002

Name: Student_2

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 1

Enter the day number (1-7): 1

Is Student_1 (ID: 1) present today? (1 = Yes, 0 = No): 1

Is Student_2 (ID: 2) present today? (1 = Yes, 0 = No): 1

Attendance for Day 1 recorded successfully!

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 1

Enter the day number (1-7): 2

Is Student_1 (ID: 1) present today? (1 = Yes, 0 = No): 1

Is Student_2 (ID: 2) present today? (1 = Yes, 0 = No): 0

Attendance for Day 2 recorded successfully!

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 1

Enter the day number (1-7): 3

Is Student_1 (ID: 1) present today? (1 = Yes, 0 = No): 1

Is Student_2 (ID: 2) present today? (1 = Yes, 0 = No): 0

Attendance for Day 3 recorded successfully!

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 1

Enter the day number (1-7): 4

Is Student_1 (ID: 1) present today? (1 = Yes, 0 = No): 0

Is Student_2 (ID: 2) present today? (1 = Yes, 0 = No): 0

Attendance for Day 4 recorded successfully!

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 1

Enter the day number (1-7): 5

Is Student_1 (ID: 1) present today? (1 = Yes, 0 = No): 1

Is Student_2 (ID: 2) present today? (1 = Yes, 0 = No): 0

Attendance for Day 5 recorded successfully!

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 1

Enter the day number (1-7): 6

Is Student_1 (ID: 1) present today? (1 = Yes, 0 = No): 1

Is Student_2 (ID: 2) present today? (1 = Yes, 0 = No): 0

Attendance for Day 6 recorded successfully!

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 1

Enter the day number (1-7): 7

Is Student_1 (ID: 1) present today? (1 = Yes, 0 = No): 0

Is Student_2 (ID: 2) present today? (1 = Yes, 0 = No): 0

Attendance for Day 7 recorded successfully!

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 2

Student_1 (ID: 1): Total Days = 5, Bill = Rupees 500.00

Has the student paid the bill? (1 = Yes, 0 = No): 1

Student_2 (ID: 2) gets a discount of Rupees 10.00 for 6 days leave.

Student_2 (ID: 2): Total Days = 1, Bill = Rupees 90.00

Has the student paid the bill? (1 = Yes, 0 = No): 0

Bills calculated and payment status updated!

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 3

====== MONTHLY SUMMARY ======

ID Name Days Bill(Rupees) Status

---

1 Student_1 5 500.00 Paid

2 Student_2 1 90.00 Unpaid

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 4

====== UNPAID STUDENTS ======

ID: 2 | Name: Student_2 | Due: Rupees 90.00

====== HOSTEL MESS MANAGEMENT SYSTEM ======

1. Mark Attendance

2. Calculate Bills

3. Display Summary

4. Show Students with Unpaid Bills

5. Exit

Enter your choice: 5

Exiting system. Thank you!

Process returned 0

Conclusion

By developing this Hostel Mess Billing and Attendance Tracker, I learned how to use structures to store multiple pieces of related data for each student. I also understood how arrays can be used to manage repeated daily records like attendance. Overall, this project structure real-world problems into code, and use fundamental C programming concepts effectively in a practical scenario.