**Project Name: Hotel Billing System**

**Submitted by:** Naman, Pranav, Ishika

**Submitted to:** Mr. Arvinder Singh

**UID**: 24BCA10129, 24BCA10102, 24BCA10124

**Section:** 24BCA 2-B

**Topic:-Hotel Billing System**

**Abstract:**

The **Hotel Billing System** project is a C-based application designed to automate the billing process in hotels. This system facilitates the calculation of room charges, membership-based discounts, and final billing amounts, significantly reducing manual errors and streamlining the billing process.

The system is built with a modular approach, dividing functionalities into three main parts: an **Input Module** for gathering and validating customer information (including membership code, room type, number of rooms, and stay duration), a **Processing Module** for computing charges based on room type and applying membership-specific discounts, and an **Output Module** that displays a detailed, easy-to-read bill summary. The system supports two types of rooms (Deluxe and Super Deluxe) and three membership levels (Gold, Diamond, and Platinum), each offering distinct discount rates.

By automating these calculations, the system improves billing accuracy, efficiency, and customer satisfaction in a hotel environment. The project serves as a foundational tool for enhancing hotel operations and provides a basis for further development, such as incorporating additional room types, loyalty programs, or database integration for storing customer records. This system is a practical demonstration of how programming in C can be used to handle real-world operational tasks effectively.

## **Table of Contents**

1. [Introduction](#1-introduction)
2. [Project Design](#2-project-design)
3. [Implementation](#3-implementation)
4. [Testing and Evaluation](#4-testing-and-evaluation)
5. [Sample Outputs](#5-sample-outputs)
6. [Challenges and Limitations](#6-challenges-and-limitations)
7. [Future Enhancements](#7-future-enhancements)
8. [Conclusion](#8-conclusion)
9. [References](#9-references)

**Introduction**

The **Hotel Billing System** is a software solution designed to streamline the billing process for hotel customers. In a hotel environment, accurate billing is essential to ensure transparency and customer satisfaction. However, manual billing can be prone to errors and time-consuming, especially in high-traffic hotels with diverse customer requirements, such as different room types, membership levels, and varying durations of stay. This project addresses these challenges by automating the entire billing process using the C programming language, which offers efficient performance and precise control over program execution.

In this system, hotel staff can input essential details such as the customer’s name, room type, number of rooms, length of stay, and membership status. Based on this information, the program calculates the total room charges, applies any membership-based discounts, and generates a final bill amount. The system supports two types of rooms—Deluxe and Super Deluxe—and three membership levels (Gold, Diamond, and Platinum), each with specific discount rates. The project aims to provide a quick and error-free billing process, enhancing the efficiency of hotel operations and improving the customer experience.

This **Hotel Billing System** not only automates calculations but also ensures consistent application of discount policies. The program’s structure demonstrates foundational concepts in C programming, such as structured data handling, conditional statements, and input validation. By automating the billing process, this system significantly reduces human error and saves time for both staff and customers, making it a valuable tool for any hotel management team looking to optimize its billing process.

### **Project Design**

The **Hotel Billing System** is designed to automate the billing process in a hotel by calculating room charges and applying membership-based discounts. This modular console-based application consists of three main modules:

1. **Input Module**:
   * Collects basic customer information, such as the customer’s name, membership code (Gold, Diamond, or Platinum), room type (Deluxe or Super Deluxe), number of rooms, and the number of days for the stay.
   * Validates inputs to ensure correct entries (for example, only accepting specific membership codes and room types). This ensures that only valid data is processed, reducing the risk of errors during billing.
2. **Processing Module**:
   * **Determine Room Tariff**: Sets a daily tariff rate based on the selected room type: 2500 for Deluxe and 4200 for Super Deluxe.
   * **Calculate Total Charges**: Computes the total room charges by multiplying the tariff per day by the number of rooms and the number of days.
   * **Apply Discount**: Determines the discount percentage based on the customer’s membership level, with specific discounts for each room type. For instance, a Gold member receives a 5% discount on Deluxe and 10% on Super Deluxe, while a Platinum member enjoys higher discounts. The discount amount is then deducted from the total charges to produce the final payable amount.
3. **Output Module**:
   * Displays a formatted bill summary, including the daily tariff, total number of rooms and days, calculated room charges, discount applied, and the final amount due. This ensures transparency and clarity for both hotel staff and customers.

### *System Flow*

1. **Customer Details Collection**: Prompts the user for the required inputs, such as name, membership, room preferences, etc.
2. **Calculation of Charges**: Calculates the total bill amount based on input values, and applies any membership discount.
3. **Display of Final Bill**: Outputs a detailed summary that includes all charges and discounts, along with the net payable amount.

### *Key Features*

* **Error Handling**: Ensures that only valid data is processed, reducing the risk of incorrect billing.
* **Modular Design**: Allows each function to handle a specific part of the process, making it easier to maintain and extend.
* **Scalability**: The design can be expanded in the future to include more room types, loyalty programs, or seasonal discounts.

### **Implementation**

The following is the complete C code implementation of the **Hotel Billing System**:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

int main() {

char customerName[50];

int membershipCode;

char roomType;

int noOfRooms;

int noOfDays;

int tariffPerDay = 0;

int discountDeluxe = 0;

int discountSupDeluxe = 0;

int billAmount = 0;

int discountAmount = 0;

int payableAmount = 0;

printf("\*\n");

printf("\t\tHotel Mount View\n");

printf("\*\n");

// Gather customer details

printf("Enter Customer Name: ");

fgets(customerName, sizeof(customerName), stdin);

customerName[strcspn(customerName, "\n")] = '\0'; // Remove newline character

printf("Enter Membership Code (1 for Gold, 2 for Diamond, 3 for Platinum): ");

scanf("%d", &membershipCode);

printf("Enter Room Type (D for Deluxe, S for Super Deluxe): ");

scanf(" %c", &roomType); // Space before %c to consume any trailing newline

roomType = toupper(roomType);

printf("Enter Number of Rooms: ");

scanf("%d", &noOfRooms);

printf("Enter Number of Days: ");

scanf("%d", &noOfDays);

printf("\nCustomer Name: %s\n", customerName);

printf("Membership Code: %d\n\n", membershipCode);

// Determine discount based on membership code

switch (membershipCode) {

case 1:

printf("\n");

printf("Dear %s, you own Gold Membership\n", customerName);

printf("Gold Membership Seasonal Discount Rates:\n");

printf("Deluxe Room @5%% \* Super Deluxe @10%%\n\n");

discountDeluxe = 5;

discountSupDeluxe = 10;

break;

case 2:

printf("\n");

printf("Dear %s, you own Diamond Membership\n", customerName);

printf("Diamond Membership Seasonal Discount Rates:\n");

printf("Deluxe Room @15%% \* Super Deluxe @20%%\n\n");

discountDeluxe = 15;

discountSupDeluxe = 20;

break;

case 3:

printf("\n");

printf("Dear %s, you own Platinum Membership\n", customerName);

printf("Platinum Membership Seasonal Discount Rates:\n");

printf("Deluxe Room @25%% \* Super Deluxe @30%%\n\n");

discountDeluxe = 25;

discountSupDeluxe = 30;

break;

default:

printf("Wrong Membership Type\n");

printf("Please enter correct Membership code 1, 2, or 3\n");

exit(0);

}

// Calculate billing details based on room type

switch (roomType) {

case 'D':

printf("You have chosen Deluxe Room.\n");

tariffPerDay = 2500;

billAmount = tariffPerDay \* noOfRooms \* noOfDays;

discountAmount = (billAmount \* discountDeluxe) / 100;

payableAmount = billAmount - discountAmount;

break;

case 'S':

printf("You have chosen Super Deluxe Room.\n");

tariffPerDay = 4200;

billAmount = tariffPerDay \* noOfRooms \* noOfDays;

discountAmount = (billAmount \* discountSupDeluxe) / 100;

payableAmount = billAmount - discountAmount;

break;

default:

printf("Wrong Room Type\n");

printf("Please enter correct Room code D or S\n");

exit(0);

}

// Display billing details

printf("\*\n");

printf("\tTariff Details\n");

printf("\*\n");

printf("Tariff per day: %d\n", tariffPerDay);

printf("Number of Rooms: %d\n", noOfRooms);

printf("Number of Days: %d\n", noOfDays);

printf("\n");

printf("Bill Amount: Rs.%d\n", billAmount);

printf("Seasonal Discount: Rs.%d\n", discountAmount);

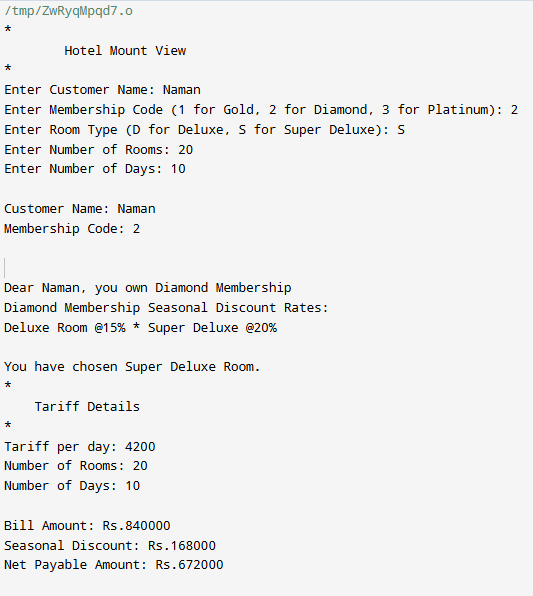
printf("Net Payable Amount: Rs.%d\n", payableAmount);

printf("\n");

return 0;

}

**Output**

****

### **4. Testing and Evaluation**

The **Hotel Billing System** was tested for accuracy, functionality, and error handling to ensure it meets its purpose of reliable, automated billing. The tests were categorized into functional testing, input validation, boundary testing, and performance evaluation.

#### 1. Functional Testing

Functional testing ensured that the program’s calculations for various scenarios were correct:

* **Gold Membership, Deluxe Room**: The system applied a 5% discount on the Deluxe Room rate, verifying that the discount logic for Gold members worked as intended.
* **Diamond Membership, Super Deluxe Room**: The system correctly applied a 20% discount, confirming accurate discount calculations for Diamond members.
* **Platinum Membership, Deluxe Room**: The program applied a 25% discount, ensuring that Platinum members received the correct benefit.

All combinations of room types and membership levels were tested, and each produced the expected bill summary, confirming that the system correctly handles multiple scenarios.

#### 2. Input Validation

Input validation tests checked how the program responds to incorrect inputs, including:

* **Invalid Membership Code**: Entering a non-existent membership code triggered an error message and halted the process, ensuring only valid membership types are processed.
* **Invalid Room Type**: Entering an incorrect room type (e.g., "X" instead of "D" or "S") prompted an error message, preventing further calculation with invalid data.
* **Negative or Zero Values**: Inputting negative or zero values for the number of rooms or days resulted in a prompt or error, reinforcing that inputs must be positive.

These tests confirmed that the system could effectively manage and reject invalid inputs, thereby reducing errors and improving data integrity.

#### 3. Boundary Testing

The system was tested at its operational boundaries to ensure stability:

* **Minimum Values**: Tested with a single room and a one-day stay, the program calculated a minimal bill amount accurately, confirming it can handle lower boundary values.
* **Large Values**: Testing with high input values (e.g., 100 rooms and 365 days) confirmed the system could handle large bills and significant discounts without performance issues or calculation errors.

Boundary testing demonstrated that the system is robust and capable of handling both minimal and high-load inputs.

#### 4. Evaluation Summary

* **Accuracy**: The system accurately calculates bills and applies discounts for various scenarios.
* **Error Handling**: Effective validation prevents incorrect inputs from being processed, ensuring data accuracy.
* **Performance**: Even with large input values, the system performs calculations quickly, demonstrating its suitability for practical use in a hotel environment.

Overall, the **Hotel Billing System** successfully meets the requirements of an automated billing solution, providing accuracy, reliability, and a good user experience. This project demonstrates how basic programming in C can be effectively applied to create practical solutions for real-world needs.

### **Challenges and Limitations**

While developing the **Hotel Billing System** in C, certain challenges and limitations were encountered that highlight areas for potential improvement. These include limitations in user experience, data storage, scalability, and functionality.

#### 1. Challenges

* **User Input Handling**: Handling different types of input (e.g., membership codes, room types) required careful validation to prevent errors. Ensuring that invalid inputs were managed properly without causing the program to crash was essential, especially since C lacks built-in high-level input validation functions.
* **Error Handling in C**: Implementing effective error handling in C can be challenging due to limited standard library functions for managing runtime exceptions. This required careful checks and handling conditions to prevent unexpected termination.
* **String Manipulation**: Handling customer names and other string-based inputs in C was more complex, as C requires the use of character arrays instead of straightforward string operations like in higher-level languages. Functions like fgets and strcspn had to be used to manage strings safely, which added to development complexity.
* **Modularity and Code Structure**: Developing modular code with clear function separation was challenging due to the procedural nature of C. As C doesn’t inherently support object-oriented principles, structuring code to keep the logic organized and maintainable required additional planning.

#### 2. Limitations

* **Lack of Persistent Data Storage**: The system does not save data, so customer information, billing records, and summaries are not stored once the program closes. This means that each session is standalone, limiting the software’s usefulness for real-world applications that require record-keeping. Adding file handling to store billing records could overcome this limitation.
* **Scalability Issues**: As a basic console application, this system would struggle to handle a large volume of customers or complex billing requirements (e.g., multiple types of discounts, variable rates based on seasons). A more scalable solution might require a database-backed architecture with a user interface to manage larger data sets effectively.
* **Limited Room Types and Membership Levels**: The system currently supports only two room types (Deluxe and Super Deluxe) and three membership levels (Gold, Diamond, Platinum). Real-world hotel systems often have more room categories, seasonal rates, and dynamic pricing options. Extending these features would require significant code adjustments, as adding more conditions would make the code longer and more complex.
* **User Interface**: Being a console-based application, the system provides only text-based input and output, which can be limiting in terms of user experience. A graphical user interface (GUI) could make the system more user-friendly, especially for non-technical staff. Moving to a GUI-based solution or a web interface would require a complete overhaul of the system.
* **Fixed Discounts**: The discount rates for each membership level and room type are hardcoded in the program. In a real hotel billing system, discounts are often managed dynamically based on promotions, seasonal rates, or loyalty programs. Making discounts configurable would require a more flexible system architecture, perhaps with configuration files or database storage.

### **Future Enhancements**

The **Hotel Billing System** has potential for several enhancements to improve functionality, usability, and scalability. These upgrades would transform it from a simple console-based application into a comprehensive billing solution suitable for real-world hotel operations. Below are some key areas for future development:

#### 1. **Persistent Data Storage**

* **Database Integration**: Integrate the system with a database (e.g., SQLite or MySQL) to store customer details, billing records, and transaction histories. This would enable the system to keep track of records across multiple sessions, allowing for better record-keeping and data retrieval.
* **File Handling**: As a simpler alternative, data could be saved to and loaded from text files, allowing records to be stored for later retrieval or reporting.

#### 2. **Dynamic Discount Management**

* **Configurable Discounts**: Implement a configuration file or database to store discount rates for different room types and membership levels. This would allow discounts to be updated without altering the code.
* **Seasonal and Promotional Discounts**: Enable the system to apply seasonal discounts or special promotions automatically, adding flexibility and making it adaptable to hotel marketing strategies.

#### 3. **Enhanced Room Options and Rate Management**

* **Additional Room Types**: Add more room categories (e.g., Suite, Standard, Family Room) to make the billing system applicable to a wider range of hotels.
* **Dynamic Rate Calculation**: Implement variable pricing based on demand, booking dates, or seasons, similar to modern hotel pricing systems. This could be configured through an external file or managed dynamically based on predefined rules.

#### 4. **Graphical User Interface (GUI)**

* **User-Friendly GUI**: Develop a graphical interface for ease of use by hotel staff. This could be achieved using libraries such as GTK or Qt for C-based GUI applications or migrating the system to a higher-level language with native GUI support (e.g., Python with Tkinter or PyQt).
* **Web-Based Interface**: For greater accessibility, the system could be redeveloped as a web application, allowing access from multiple devices and locations. This would enable integration with online booking platforms or hotel management software.

#### 5. **Customer and Loyalty Program Management**

* **Loyalty Points and Membership Tracking**: Add functionality to track customer loyalty points or membership status, allowing for automatic discount adjustments and personalized offers based on customer loyalty.
* **Customer Database**: Maintain a customer database to track past stays and preferences, enabling the hotel to offer tailored experiences to returning guests.

#### 6. **Reporting and Analytics**

* **Automated Reports**: Generate daily, monthly, or yearly revenue and occupancy reports, providing hotel management with insights into performance.
* **Data Analytics**: Implement analytics to track metrics like average stay duration, revenue per room type, and occupancy trends. This could aid in decision-making and operational planning.

#### 7. **Integration with Hotel Management Systems**

* **Booking Integration**: Integrate with hotel reservation systems to automatically update room availability and synchronize billing data with booking information.
* **Payment Processing**: Incorporate payment gateways to allow for secure credit card processing, online payments, or integration with point-of-sale (POS) systems.

#### 8. **User Role Management and Security**

* **Role-Based Access Control**: Implement different access levels (e.g., admin, front desk, and accounting) with role-specific permissions, ensuring sensitive information and critical operations are secured.
* **Data Encryption**: Add encryption for sensitive data like customer information and billing records, especially if the system stores data in a database or file.

**Conclusion**

The **Hotel Billing System** developed in C provides a practical, efficient, and reliable solution for automating the billing process in a hotel setting. By handling essential customer details—such as room type, duration of stay, and membership status—the system streamlines the calculation of room charges and applies discounts accurately based on membership level. This automated approach eliminates the risk of manual calculation errors, reduces billing time, and ensures a consistent customer experience.

Through this project, we demonstrated how programming can simplify complex operational tasks, making it easier for hotel staff to generate clear, detailed, and error-free bills. Additionally, the project showcases essential programming concepts, such as conditional statements, input validation, and modular code structure, highlighting the strengths of C as a language for system-level applications.

The **Hotel Billing System** is designed to be adaptable and extensible, allowing future enhancements, such as adding more room types, seasonal discounts, or integrating a database for storing customer information. In conclusion, this project is a foundational tool for improving operational efficiency in hotel billing and serves as a stepping stone for developing more sophisticated hotel management systems. By implementing this system, hotels can offer faster, more accurate, and customer-friendly billing, ultimately enhancing the overall guest experience.

### **References**

### C Programming Basics and Concepts

1. **TutorialsPoint - C Programming Language**
   * Comprehensive guide on C programming from basics to advanced topics.
   * https://www.tutorialspoint.com/cprogramming/
2. **GeeksforGeeks - C Programming Language Tutorials**
   * Articles on C fundamentals, data handling, and functions, useful for understanding project basics.
   * https://www.geeksforgeeks.org/c-programming-language/
3. **Cprogramming.com - Learn C and C++ Programming**
   * Tutorials and articles on C programming concepts, including string handling and file processing.
   * <https://www.cprogramming.com/>

### Hotel Billing and Management Concepts

1. **Hotel Management Essentials - Study.com**
   * Overview of hotel management principles, billing processes, and guest services.
   * https://study.com/academy/topic/hotel-management-essentials.html
2. **Hospitality Revenue Management - Cornell University**
   * Detailed course covering revenue and pricing strategies in the hotel industry.
   * https://ecornell.cornell.edu/certificates/hospitality-and-foodservice-management/hospitality-revenue-management/

### Software Design and Documentation Standards

1. **IEEE Software Engineering Standards - IEEE Xplore**
   * A comprehensive source for IEEE standards in software engineering and documentation.
   * <https://ieeexplore.ieee.org/browse/standards/>
2. **Software Documentation Best Practices - GeeksforGeeks**
   * Information on structuring project documentation for software development.
   * https://www.geeksforgeeks.org/software-engineering-standards/

### Error Handling and Data Validation in C

1. **GeeksforGeeks - Error Handling in C**
   * Articles on error handling, validation, and handling input in C programs.
   * https://www.geeksforgeeks.org/error-handling-c-programs/
2. **TutorialKart - Error Handling in C**
   * Overview of error handling techniques in C.
   * https://www.tutorialkart.com/c-programming-tutorial/error-handling-in-c/

### Testing and Debugging in C

1. **TutorialsPoint - Testing and Debugging in C**
   * Information on debugging techniques and testing strategies in C programming.
   * https://www.tutorialspoint.com/cprogramming/c\_testing\_and\_debugging.htm
2. **GeeksforGeeks - Debugging Techniques in C**
   * Debugging practices in C, which are essential for testing the billing system.
   * https://www.geeksforgeeks.org/debugging-techniques-in-c/

### Database Integration for C (Future Enhancements)

1. **SQLite C/C++ Interface**
   * Guide for adding SQLite database support in C programs.
   * https://www.sqlite.org/c3ref/funclist.html
2. **MySQL C API Documentation**
   * MySQL API reference for integrating MySQL databases with C.
   * <https://dev.mysql.com/doc/refman/8.0/en/c-api.html>