S.No: 1	Exp. Name: ***Project Module***	**Date: 2024-06-14**

Aim:

Project Module

Source Code:

```
hello.c
```

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

#define MAX_TABLES 30

typedef struct FoodItem {
    char name[30];
    float price;
    int quantity;
    struct FoodItem* next;
} FoodItem;

typedef struct Table {
    int tableNumber;
    int isOccupied;
} Table;

// Function prototypes
FoodItem* createFoodItem(char* name, float price, int quantity);
void appendFoodItem(FoodItem** head, char* name, float price, int
quantity);
void viewMenu(FoodItem* head);
void checkAvailability(FoodItem* head, char* foodName);
void orderFood(FoodItem* head, int index, int quantity);
void initializeMenu(FoodItem** menu);
void freeMenu(FoodItem* head);
void initializeTables(Table tables[]);
int findEmptyTable(Table tables[]);
void occupyTable(Table tables[], int tableNumber);
void freeTable(Table tables[], int tableNumber);

// Function to create a new food item
FoodItem* createFoodItem(char* name, float price, int quantity) {
    FoodItem* newItem = (FoodItem*)malloc(sizeof(FoodItem));
    strcpy(newItem->name, name);
    newItem->price = price;
    newItem->quantity = quantity;
    newItem->next = NULL;
    return newItem;
}

// Function to append a food item to the menu
void appendFoodItem(FoodItem** head, char* name, float price, int
quantity) {
    FoodItem* newItem = createFoodItem(name, price, quantity);
```

ID: 2303811710621114
2023-2027-L
K.Ramakrishnan College of Technology

```
    if (*head == NULL) {
        *head = newItem;
        return;
    }
    FoodItem* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newItem;
}

// Function to view the menu
void viewMenu(FoodItem* head) {
    FoodItem* temp = head;
    int index = 0;
    printf("Menu:\n");
    while (temp != NULL) {
        printf("%d. %s - $%.2f - Available: %d\n", index, temp-
>name, temp->price, temp->quantity);
        temp = temp->next;
        index++;
    }
}

// Function to check the availability of a food item by name
void checkAvailability(FoodItem* head, char* foodName) {
    FoodItem* temp = head;
    while (temp != NULL) {
        if (strcmp(temp->name, foodName) == 0) {
            printf("%s - $%.2f - Available: %d\n", temp->name,
temp->price, temp->quantity);
            return;
        }
        temp = temp->next;
    }
    printf("Food item '%s' not found in menu.\n", foodName);
}

// Function to order a food item by index
void orderFood(FoodItem* head, int index, int quantity) {
    FoodItem* temp = head;
    int currentIndex = 0;
    while (temp != NULL) {
        if (currentIndex == index) {
            if (temp->quantity >= quantity) {
                temp->quantity -= quantity;
```

K.Ramakrishnan College of Technology 2023-2027-L

```
                printf("Order placed for %d %s(s). Remaining:
%d\n", quantity, temp->name, temp->quantity);
            } else {
                printf("Not enough %s available. Available:
%d\n", temp->name, temp->quantity);
            }
            return;
        }
        temp = temp->next;
        currentIndex++;
    }
    printf("Food item at index '%d' not found in menu.\n",
index);
}

// Function to initialize the menu with default items
void initializeMenu(FoodItem** menu) {
    appendFoodItem(menu, "Burger", 5.99, 10);
    appendFoodItem(menu, "Pizza", 8.99, 5);
    appendFoodItem(menu, "Pasta", 7.99, 7);
    appendFoodItem(menu, "Salad", 4.99, 12);
    appendFoodItem(menu, "Soup", 3.99, 8);
    appendFoodItem(menu, "Sandwich", 4.49, 15);
    appendFoodItem(menu, "Fries", 2.99, 20);
    appendFoodItem(menu, "Ice Cream", 3.49, 10);
    appendFoodItem(menu, "Soda", 1.99, 25);
    appendFoodItem(menu, "Coffee", 2.49, 18);
}

// Function to free the allocated memory for the menu
void freeMenu(FoodItem* head) {
    FoodItem* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

// Function to initialize tables
void initializeTables(Table tables[]) {
    for (int i = 0; i < MAX_TABLES; i++) {
        tables[i].tableNumber = i + 1;
        tables[i].isOccupied = 0;
    }
```

```c
// Function to find an empty table
int findEmptyTable(Table tables[]) {
    for (int i = 0; i < MAX_TABLES; i++) {
        if (!tables[i].isOccupied) {
            return tables[i].tableNumber;
        }
    }
    return -1;
}

// Function to occupy a table
void occupyTable(Table tables[], int tableNumber) {
    if (tableNumber > 0 && tableNumber <= MAX_TABLES) {
        tables[tableNumber - 1].isOccupied = 1;
    }
}

// Function to free a table
void freeTable(Table tables[], int tableNumber) {
    if (tableNumber > 0 && tableNumber <= MAX_TABLES) {
        tables[tableNumber - 1].isOccupied = 0;
    }
}

// Main function
int main() {
    FoodItem* menu = NULL;
    initializeMenu(&menu);

    Table tables[MAX_TABLES];
    initializeTables(tables);

    int choice;
    char foodName[30];
    int index;
    int quantity;

    while (1) {
        printf("\nHotel Management System\n");
        printf("1. View Menu\n");
        printf("2. Check Food Availability\n");
        printf("3. Order Food\n");
        printf("4. Find an Empty Table\n");
        printf("5. check a favour Table\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar(); // Consume newline character

        switch (choice) {
            case 1:
                viewMenu(menu);
                break;
            case 2:
                printf("Enter food name to check availability: 
");
                fgets(foodName, sizeof(foodName), stdin);
                foodName[strcspn(foodName, "\n")] = '\0'; // 
Remove trailing newline
                checkAvailability(menu, foodName);
                break;
            case 3:
                printf("Enter menu index to order: ");
                scanf("%d", &index);
                printf("Enter quantity to order: ");
                scanf("%d", &quantity);
                orderFood(menu, index, quantity);
                break;
            case 4: {
                int tableNumber = findEmptyTable(tables);
                if (tableNumber != -1) {
                    printf("Empty table found: Table %d\n", 
tableNumber);
                    occupyTable(tables, tableNumber);
                } else {
                    printf("No empty tables available.\n");
                }
                break;
            }
            case 5: {
                int tableNumber;
                printf("Enter table number to free: ");
                scanf("%d", &tableNumber);
                freeTable(tables, tableNumber);
                printf("Table %d is now free.\n", tableNumber);
                break;
            }
            case 6:
                printf("Exiting...\n");
                freeMenu(menu);
                return 0;
        }
    }

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

```
Hello World
```

Info

Lab record generated as PDF successfully

OK