DESIGN AND MANAGE DEPARTMENT CALENDAR

{BATCH INCHARGE: M/S PRITI WARUNGSE MAM}

GROUP MEMBERS		
SR.NO	PRN	NAMES
1.	B24IT1056	ATHARVA PARDESHI
2.	B24IT1020	OMKAR MARKAD
3.	B24IT1005	KARAN BHAPKAR
4.	B24IT1002	DEEPAK ILLPATE

OCTOBER 28, 2024

ENGG AND EXPLORATION LAB{EEL}

MARATHWADA MITRA MANDAL COLLEGE OF ENGG PUNE.

PROJECT-1

PROGRAMMING AND LOGIC BUILDING...

TOPIC :- DESIGN AND MANAGE DEPARTMENT CALENDAR.

➢ Introduction

In today's fast-paced world, managing time effectively is crucial. A calendar serves as an essential tool for organizing our schedules, tracking events, and celebrating special occasions. While traditional calendars provide a simple view of days, months, and years, incorporating events such as festivals can enhance their utility and make them more meaningful.

This project focuses on creating a calendar program in C that not only displays the days of a specific month and year but also highlights important festivals and events. By combining basic calendar functionalities with the ability to showcase cultural and national celebrations, this program aims to offer a more interactive and informative experience.

The calendar program is designed to:

- Prompt users for a specific month and year.
- Calculate the starting day of that month.
- Adjust for leap years to ensure accurate day counts.
- Display the calendar layout with the days of the month aligned correctly.
- Include a list of predefined festivals, indicating their occurrence within the calendar.

Through this implementation, users will gain a better appreciation of how programming can intersect with everyday life, enabling them to plan and celebrate important dates more effectively. The program serves as a foundation that can be further expanded with features such as user-defined events, reminders, or even integration with digital calendars.

As we delve into the details of the program, we will explore the logic behind calendar calculations, the handling of user input, and the incorporation of events, all while emphasizing the importance of clear and maintainable code.

➢ DESIGN AND ANALYSIS

Creating a design for your calendar program involves outlining the structure, components, and flow of the program. Here's a design plan that breaks down the key elements:

Design Outline for Calendar Program with Festivals

1. Program Structure

- Header Files: Include necessary libraries such as <stdio.h> and <string.h>.
- Data Structures:
  - Define a Festival structure to hold information about festivals.
- Function Prototypes:
  - int get Starting Day(int year, int month);
  - int is Leap Year (int year);

2. Data Structures

- Festival Structure:

```
typedef struct {
    int day;            // Day of the festival
    int month;          // Month of the festival
    char name[50];      // Name of the festival
} Festival;
```

- **Array of Festivals**:
  - An array to hold multiple festival entries.

3. Main Function Flow

- **Input Handling**:
  - Prompt the user to enter a month and a year.
  - Validate the input to ensure it is within acceptable ranges.

- **Leap Year Adjustment**:
    - Check if the entered year is a leap year and adjust February's days accordingly.
- **Calculate Starting Day**:
    - Call get Starting Day () to find out which day of the week the month starts on.
- **Print Calendar**:
    - Print the header for the calendar.
    - Print leading spaces based on the starting day.
    - Loop through the days of the month and check for festivals.
    - Print each day and indicate if there's a festival.

4. Functions

- **Get Starting Day (int year, int month)**:
    - Calculate the day of the week for the first day of the month using Zeller's Congruence.
- **Is Leap Year (int year)**:
    - Determine if the specified year is a leap year.

5. Festival Handling

- **Predefined Festivals**:
    - Create a static array of festivals with specific dates and names.
- **Display Festivals**:
    - While printing each day, check if it matches a festival date and mark it accordingly.

➢ IMPLEMENTATION

➢ Here's the complete implementation of a calendar program in C that includes festival functionality, based on the design outlined earlier. This program allows the user to enter a month and year, calculates the starting day, and displays the calendar while highlighting specific festivals.

1. **Our Complete C Program**

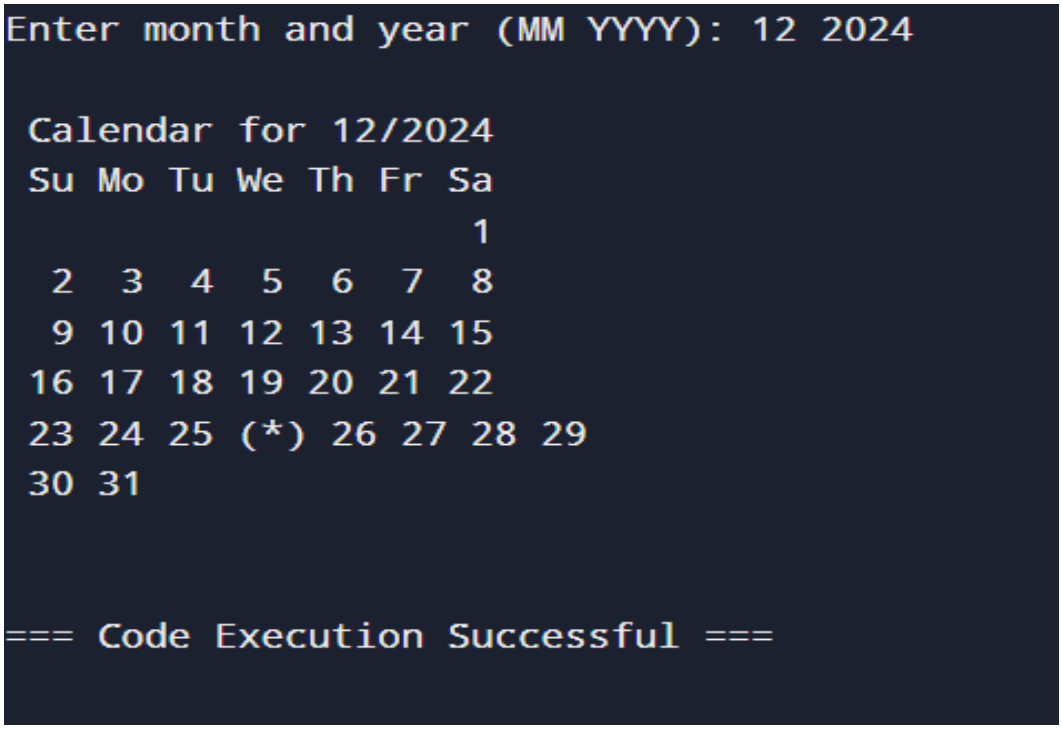
```c
#include <stdio.h>
#include <string.h>

#define MAX_FESTIVALS 10

typedef struct {
    int day;            // Day of the festival
    int month;          // Month of the festival
    char name[50];      // Name of the festival
} Festival;

// Function prototypes
int getStartingDay(int year, int month);
int isLeapYear(int year);
```

```c
int main() {
    int month, year, daysInMonth;
    int startDay;

    // Array of days in each month
    int daysInMonths[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    // Define some festivals
    Festival festivals[MAX_FESTIVALS] = {
        {1, 1, "New Year's Day"},
        {14, 2, "Valentine's Day"},
        {25, 12, "Christmas"},
        // Add more festivals as needed...
    };
    int festivalCount = 3; // Adjust based on the number of festivals

    // Input loop for validation
```

```c
while (1) {
    printf("Enter month and year (MM YYYY): ");
    if (scanf("%d %d", &month, &year) != 2 || month < 1 || month > 12 || year < 1) {
        printf("Invalid input. Please enter a valid month (1-12) and a positive year.\
        while (getchar() != '\n'); // clear input buffer
        continue;
    }
    break;
}

// Check for leap year and adjust February days
if (isLeapYear(year)) {
    daysInMonths[1] = 29;
}

// Determine the starting day of the month
startDay = getStartingDay(year, month - 1);
daysInMonth = daysInMonths[month - 1];

printf("\n Calendar for %02d/%d\n", month, year);
printf(" Su Mo Tu We Th Fr Sa\n");

// Print leading spaces
for (int i = 0; i < startDay; i++) {
    printf("   "); // Adjusted for better alignment
}

// Print days of the month
for (int day = 1; day <= daysInMonth; day++) {
    printf("%3d", day);

    // Check if there's a festival on this date
    for (int i = 0; i < festivalCount; i++) {
        if (festivals[i].day == day && festivals[i].month == month) {
```

```
                printf(" (*)"); // Indicate a festival
                break;
            }
        }

        if ((day + startDay) % 7 == 0) {
            printf("\n");
        }
    }
    printf("\n");
    return 0;
}

// Function to determine the starting day of the month
int getStartingDay(int year, int month) {
    int day = 1;
    int century = (year / 100);
    int y = year % 100;

    if (month < 3) {
        month += 12;
        y -= 1;
    }
    int k = day + (13 * (month + 1)) / 5 + y + y / 4 + century / 4 - 2 * century;
    return (k + 7) % 7;
}

// Function to check if the year is a leap year
int isLeapYear(int year) {
    return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
}
```

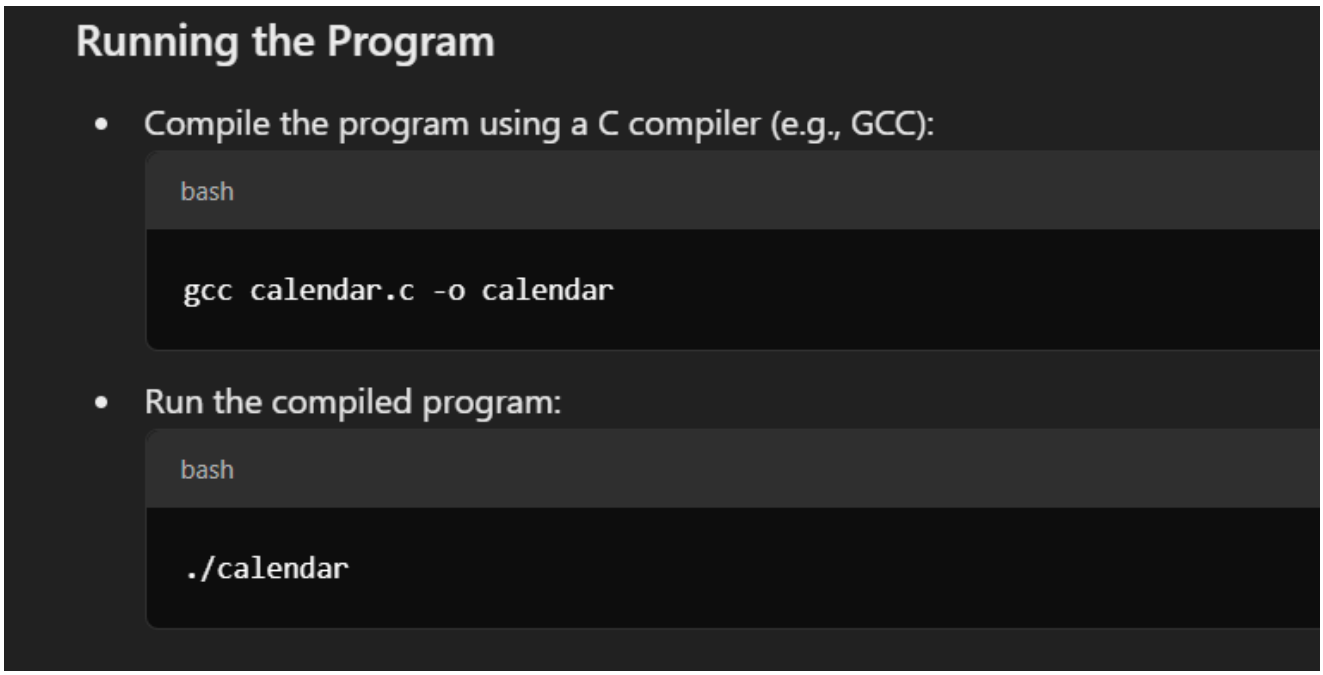
OUTPUT



Explanation of the Code

1. **Data Structures**:
    - Festival structure holds information about festivals, including the day, month, and name.
2. **Input Handling**:
    - The program prompts the user to enter a month and year and includes validation to ensure the inputs are within valid ranges.
3. **Leap Year Calculation**:
    - The program checks if the entered year is a leap year and adjusts the number of days in February accordingly.
4. **Starting Day Calculation**:
    - The function getStartingDay uses a modified Zeller's Congruence to determine the starting day of the month.
5. **Calendar Printing**:
    - The program prints the calendar header, leading spaces based on the starting day, and each day of the month.

- It checks if each day corresponds to a festival and marks it with an asterisk ((*)).



➢ Conclusion

The provided C program effectively generates a calendar for a specified month and year, highlighting important festivals. By leveraging a structured approach with a Festival data type and employing functions for calculating the starting day of the month and checking for leap years, the program is both organized and efficient.

Key features include:

Input Validation: Ensures that users enter a valid month and year, enhancing robustness.

Leap Year Calculation: Adjusts the number of days in February based on whether the year is a leap year, ensuring accurate calendar representation.

Festival Highlighting: Displays festivals on their respective dates, making the calendar more informative.

Potential enhancements could involve allowing users to add their own festivals, improving the output formatting, and expanding the program to handle additional functionalities, such as viewing multiple months or generating annual calendars.

Overall, this program serves as a solid foundation for creating a more comprehensive calendar application, with opportunities for further development and refinement.

THANKYOU.....