**UNIVERSITY INSTITUTE OF COMPUTING**

**PROJECT REPORT**

**ON**

**CALANDER**

Program Name: BCA

Subject Name/Code: Data Structures(23CAT-201)

**Submitted by: Submitted to:**

**Name:** Prabin Kumar  **Name:** Mrs. Shilpi Mittal

**UID:** 23BCA10560 **Designation:** Co-ordinator

**Section:** BCA – 3 “A”

**Group:** 6

ABSTRACT

Introduction:

**This C-based Calendar Management System is designed to display yearly calendars, showing each month’s layout in a tabular form with days aligned in columns. The program accounts for leap years, ensuring February has an extra day when appropriate. The system also dynamically calculates the starting weekday for each month based on the specified year, creating an efficient and organized calendar display.**

Technique:

The application is developed in C, employing functions and modular programming principles to achieve a structured and reusable codebase. Key techniques include:

* **Modular Design**: Individual functions are used to handle tasks such as calculating the first day of the year, checking for leap years, and printing calendar months.
* **File Management**: The program writes calendar outputs to a file, making it accessible for review or printing.
* **Date Calculations**: Leap year checking and day-of-week calculations use mathematical logic to determine correct month lengths and starting days.

System Configuration:

* **OS:** Windows 11 Home Single Language
* **Processor:** AMD Ryzen 7 5800HS with Radeon Graphics 3.20 GHz
* **RAM:** 16.0 GB; 8 GB recommended
* **Development Environment:** Any C++ IDE (e.g., Visual Studio, Code::Blocks) or Visual Studio Code with a C++ compiler (GCC or Microsoft C++ Compiler)

SUMMARY

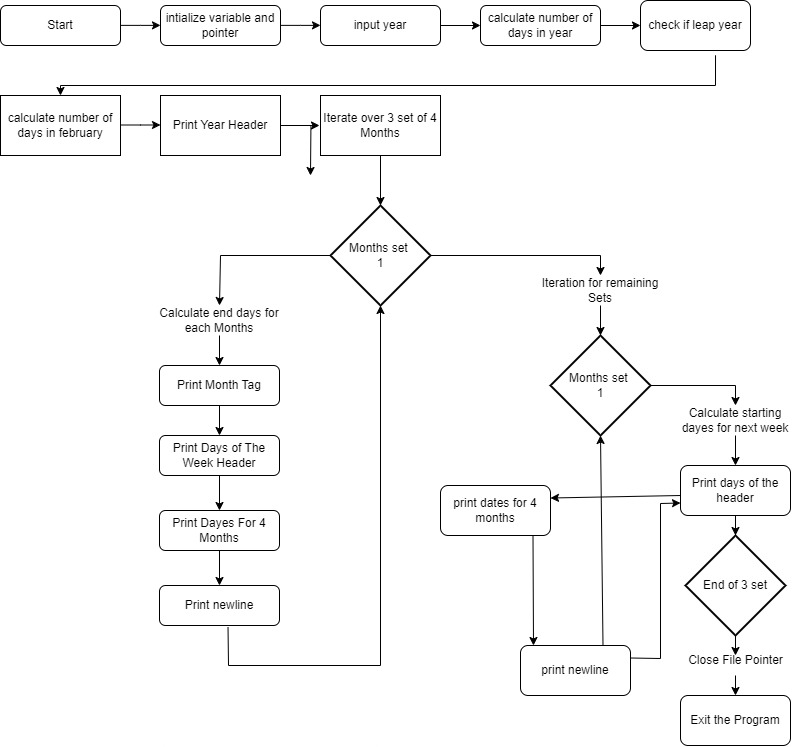
Input:

**Input**

* **User is prompted to enter a year (between 1950 and 2100).**
* **The entered year is validated and used to generate the calendar.**
* **The input year is recorded in calendarRecords.txt for future reference.**

**Process**

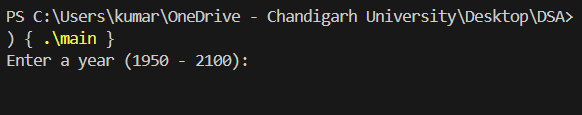
1. **Leap Year Calculation**
   * **Determines if the year is a leap year using standard leap year rules (divisible by 4, not divisible by 100, or divisible by 400).**
   * **Updates February's days to 29 if it's a leap year; otherwise, February has 28 days.**
2. **First Day of Year Calculation**
   * **Calculates the weekday of January 1st for the input year, setting the starting point for the calendar.**
   * **Uses this first day to determine the alignment of days across all months.**
3. **Monthly Layout Generation**
   * **Divides the calendar into three groups of four months each (January–April, May–August, September–December).**
   * **For each month:**
     + **Calculates initial weekday alignment and prints dates in rows of seven to match weekdays (Monday to Sunday).**
     + **Automatically moves to the next month, aligning each one accurately.**
4. **File Output**
   * **Saves the full calendar output, including month names, weekdays, and dates, in calendarRecords.txt.**
   * **Each year’s data is appended, creating a cumulative record for easy access to multiple years**



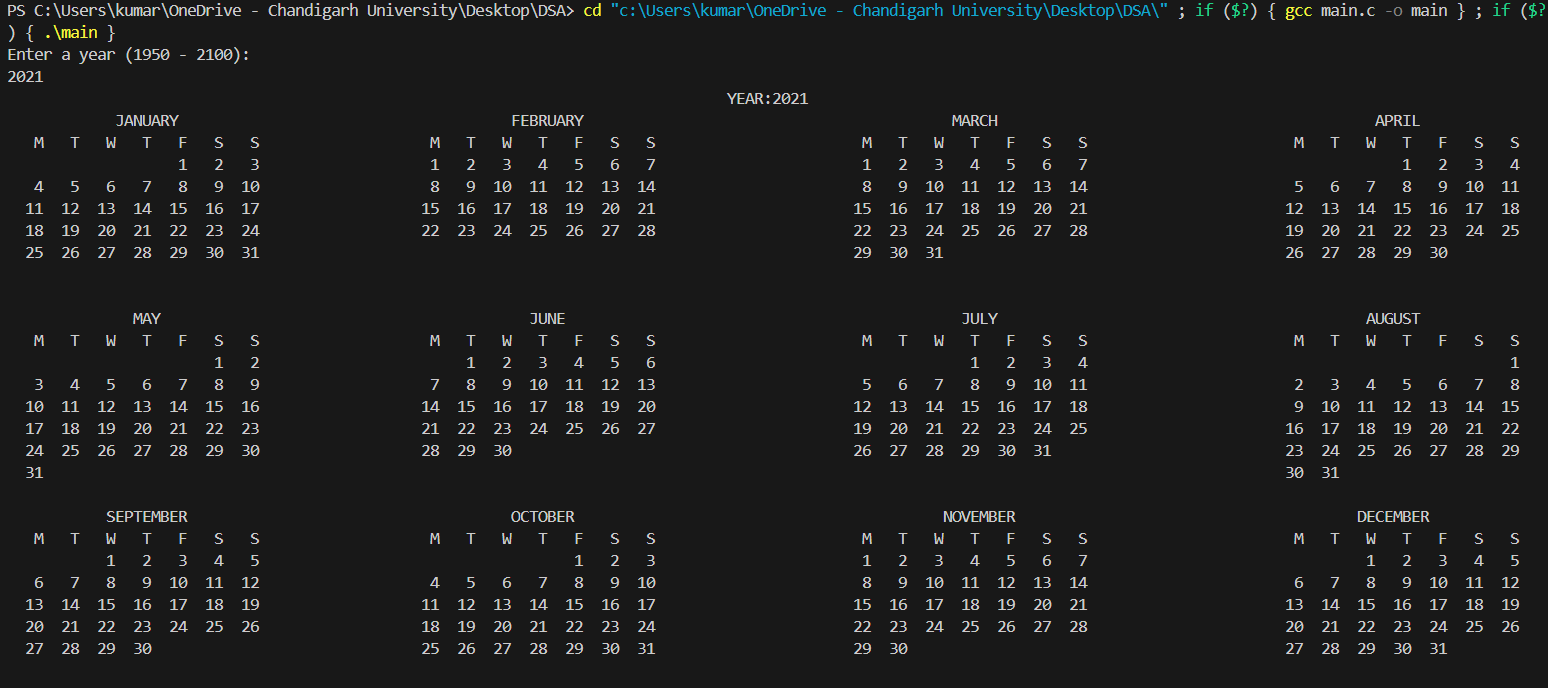
Process:

Output:

Main Menu:



User Define output:



Leap year output:

