

PROJECT REPORT

Academic Calendar in C

Course Code: CSE 100

Course Title: Software Development Project with C



Submitted By:

1. Nusratul Islam Mim (Roll: 23102015, Registration no: 11871)
2. Surma Akter Eity (Roll: 23102019, Registration no: 11875)

Department of Computer Science and Engineering

**Project Supervisor: Professor Dr. Md. Sujan Ali, Professor and Head of the
Department**

October 28, 2024

Jatiya Kabi Kazi Nazrul Islam University

Department of Computer Science and Engineering

Trishal, Mymensingh



BONAFIDE CERTIFICATE

This is to certify that the Mini Project entitled “**Academic Calendar in C**”, submitted by **Nusratul Islam Mim (Roll: 23102015)** and **Surma Akter Eity (Roll: 23102019)**, is a record of genuine work carried out by the authors in partial fulfillment of the requirements for the award of the **Bachelor of Science (BSc) degree in Computer Science and Engineering** at **Jatiya Kabi Kazi Nazrul Islam University**. This project was completed during the II Semester of the 2023-2024 academic year under the guidance of **Professor Dr. Md. Sujan Ali**.

Project Guide

Professor Dr. Md. Sujan Ali

Professor and Head of the Department

Dept. Computer Science & Engineering

External

Examiner Name:

Signature

Date:

Guide Signature

Acknowledgement

We are grateful to **Prof. Dr. Md. Sujan Ali** for his tremendous guidance and persistent support. The intelligent recommendations contributed to the project's successful completion. His experience and support during the study process improved the quality of our work and broadened our comprehension of the subject. We were lucky to work under his direction and appreciate his dedication to fostering academic success.

Furthermore, we are grateful to **Prof. Mijanur Rahman**, our Structured Programming in C lecturer, for his guidance. His detailed instruction and ability to clarify complicated issues helped us understand the technical components of the project. His compassion, determination, and devotion to our learning have been invaluable. His lessons continue to resonate with us as we advance in our academic and professional careers. We are deeply grateful to both mentors for their inspiration, knowledge, and wisdom. Their amazing contributions made this project feasible.

Nusratul Islam Mim

Roll: 23102015

Surma Akter Eity

Roll: 23102019

Contents

Abstract	4
Introduction	6
Methodology	7
Objectives of the program	11
Problem Objective	13
Requirements	15
Functions	17
Design and Implementation	19
0.1 Design	19
0.2 Implementation	19
0.3 Color Coding	19
0.4 Navigation	19
0.5 Code Implementation	20
0.5.1 Header Files	20
0.5.2 Function Definitions	20
0.5.3 Main Function	21
Function Definitions	25
0.6 User Experience and Interaction	26
Testing and Validation	27
0.7 Test Cases	27
Results and Discussion	29
Conclusion	32

Abstract

Effective time management is essential for academic achievement in today's fast-paced environment. Students face multiple duties, including attending classes, completing homework, preparing for examinations, participating in extracurricular activities, and maintaining social connections. Overwhelming schedules make time management tough. As academic responsibilities increase, students must keep track of essential dates and deadlines to prevent missing assignments and events. We designed an interactive academic calendar program to help students manage their calendars and academic duties properly.

The academic calendar tool is intended to provide a simple yet effective method for students to visualize their academic commitments. The program uses an easy, user-friendly design to display a monthly calendar that highlights major academic events such as exam dates, assignment deadlines, holidays, and other notable dates in brilliant, recognizable colors. This tool allows students to distinguish between different sorts of events at a glance, boosting their ability to plan ahead and organize their time effectively. Users can alter the design of the calendar by color-coding certain events based on their preferences.

Furthermore, the program incorporates seamless navigation between months, allowing users to easily transition from one month to the next with the use of arrow keys. This ensures that students can efficiently move through different periods of their academic year to plan long-term. Whether it's planning for midterms, finals, or the next academic term, the program's smooth navigation makes it convenient for users to stay ahead of their schedule. Additionally, the program is designed to exit gracefully with the press of a key, ensuring a smooth user experience without the need for complex commands. For example, exams can be marked in red, holidays in green, and assignment deadlines in yellow. Such customization ensures that the calendar adapts to each student's unique schedule, making it an indispensable tool for time management.

The academic calendar application is very flexible beyond its basic features, giving

students the ability to modify the calendar to suit their own academic requirements. It gives users the ability to mark recurring academic deadlines, like project submissions or weekly tasks, in addition to allowing them to include personal events. Students can further improve their time management skills by creating a personalized view of their academic obligations thanks to this customisation. Additionally, the program can handle leap years and varying month lengths, guaranteeing precise date representation in any given year.

This calendar program's usefulness goes beyond time management on an individual basis. It promotes proactive planning and time allocation by providing a neat and orderly visual depiction of a student's timetable. Students who are more organized are less prone to forget deadlines or significant events, which inevitably results in improved academic performance. Students can lower stress, increase focus, and put more effort into their academics when they have a firm knowledge of their routines. Better study habits are thus encouraged, which boosts general output and achievement in both academic and extracurricular endeavors.

In conclusion, our academic calendar application offers a useful and adaptable way for students to easily manage their academic obligations. It is a complete tool for encouraging efficient time management because of its user-friendly layout, color-coded events, and simple navigation. The program's objective is to help students manage their schedules in a visually appealing way in order to increase productivity, lower stress levels, and eventually support long-term academic achievement. This project aims to enable students to effectively manage their time, make plans, and reach their maximum academic potential.

Introduction

Students frequently balance a variety of obligations in the fast-paced academic setting of today, including classes, homework, tests, and extracurricular activities. Stress and trouble remembering critical dates and deadlines might result from this complexity. We offer a creative solution to these problems in the form of an interactive academic calendar tool made especially for teachers and students.

This project's main objective is to create an easy-to-use calendar that uses an intuitive color-coding scheme to emphasize important dates, deadlines, and holidays in addition to showing the academic timetable. With the use of arrow keys, users may easily traverse through the months in this program, giving them quick access to forthcoming academic obligations and events. The 'Esc' key makes it simple to quit the application, guaranteeing a seamless and effective user experience.

In order to meet the demands of both teachers and students, this academic calendar incorporates necessary features. It enables users to efficiently manage their time by giving them a clear and well-organized perspective of their obligations. With the help of the configurable features, users may add reminders and personal events to their calendars to suit their own needs.

Additionally, the program's goal is to improve academic achievement by enabling more effective scheduling around significant events. By visually differentiating different kinds of events, such as midterm exams, assignment due dates, and holidays, the color-coded method lessens the cognitive load that comes with remembering several obligations.

In conclusion, the goal of this project is to develop a dependable and easily usable tool that gives students the ability to take charge of their academic careers. The interactive academic calendar is a useful tool for creating a well-organized and successful academic experience by reducing the strain of juggling many obligations and encouraging a balanced approach to academic success. We expect that this program will have a positive impact on kids' academic performance and general well-being.

Methodology

Building an extensive and effective calendar printing program with color-coded events is the main goal of this project's methodology. Every development phase of the project is approached in an organized manner, guaranteeing clarity and seamless execution from beginning to end. The steps listed below provide a detailed description of the development process:

1. **Requirement Analysis:** Understanding the program's needs was the first and most important stage in creating this project. This required outlining the program's main objectives and characteristics:

- The ability to generate and print a calendar for any given month and year.
- Using color coding for important dates and particular events makes it simple to distinguish between different kinds of days (holidays, assignments, off days, etc.).
- use basic user commands to switch between months (e.g., going back from June to May or from January to February).
- effectively managing particular edge circumstances, such as leap years and months with varying day lengths.

Before beginning the design process, a thorough requirement analysis guarantees that all necessary features have been found and taken into consideration, as well as aids in precisely defining the project's scope. Anticipating user interactions is another aspect of the analysis, which guarantees that the finished product is functional, intuitive, and achieves the intended goals.

2. **Design:** The program's real structure is mapped out during the design phase. Important elements of the design consist of:

- **Modularity:** The program was created with the intention of being modular, which means that distinct functions perform various duties. For instance, the

function for printing the calendar with color coding, the day-of-week calculation, and the leap year calculation are all independent functions. This guarantees scalability, ease of maintenance, and the capacity to add or change features without affecting the entire codebase.

- **User-Friendly Interface:** The user experience is prioritized during the design phase in addition to functionality. The calendar has obvious navigation choices and is made to be easy to read. An event structure, for example, keeps track of details like the date and the event's color code. This guarantees the effective retrieval and display of event data throughout the calendar print process.

Establishing the overall program architecture occurs during the crucial design phase. It guarantees that the code stays well-structured, effective, and flexible for upcoming requirements.

3. **Implementation:** Once the design was finalized, the implementation phase began, where the C language was used to bring the design to life. The following key tasks were executed during this phase:

- **Accurate Leap Year and Day-of-Week Calculations:** The program needs to determine whether the year provided by the user is a leap year and calculate the correct day of the week on which the month starts. These calculations ensure that the printed calendar is accurate regardless of the year or month.
- **Color-Coding Based on Event Inputs or User-Defined Dates:** This feature lets users designate dates that should be emphasized with particular colors, like holidays in red or noteworthy occasions in green. When the calendar is printed, these colors are used to create a visually appealing display.
- **Easy Month-to-Month Navigation:** By entering commands, the application enables users to switch between months with ease. This feature allows the user to return to a prior month or go to the next month without having to restart the application.

This stage is essential to guarantee that the program's fundamental features are met and that it not only operates correctly but also reacts to user inputs naturally, making it simple to interact with the calendar. [2].

4. **Testing:** Any software development process must include testing, and in order to confirm the program's accuracy and functioning, extensive testing was done for this project. We evaluated a number of scenarios, including:

- **Monthly Testing:** To make sure that the number of days and the beginning day for each month were accurately computed and displayed, the application was tested over a number of months and years. **Managing the Leap Year:** Particular care was taken to guarantee that February showed 28 days in non-leap years and 29 days in leap years.
- **Event Color-Coding:** To ensure that the right dates were indicated in the right colors, various event input combinations were tried.
- **User Navigation and Input:** The navigation instructions were tested extensively to make sure users could quit the software without any issues and switch between months with ease.

Testing guarantees that all features function as intended under a variety of circumstances and assists in locating any faults or flaws in the application. This guarantees that the finished product will be sturdy.

5. **Develop the App:** The full-scale development of the app is the main emphasis of the methodology's last stage. This step consists of:

- **Code Optimization:** Following functionality verification, an attempt was made to make the code more efficient. This entails reducing pointless computations and, when practical, optimizing memory utilization.
- **User-Friendly Design:** Particular attention was paid to making the final application user-friendly.

- **Feature Integration:** During this phase, features including event storage, color coding, and navigation were fully integrated and made functioning. This indicates that the application is responsive to user inputs, aesthetically pleasing, and simple to use. **Security and Scalability:** Despite being a simple terminal application, this project has been designed to accept inputs safely. Additionally, the code is organized to support future scalability. For example, it would be simple to add additional features like integrating a graphical user interface (GUI) or saving events to a file.

The end result is a calendar printing application that is completely functional, easy to use, satisfies project criteria, and operates effectively in a variety of situations.

The program has an easy-to-use interface and displays a monthly calendar with colored highlights for important academic events, holidays, and deadlines. Users can use arrow keys to traverse between months and exit the program with a single key press.

Our calendar seeks to help students improve their time management skills, productivity, and academic achievement by providing a clear and structured view of their academic commitments. The adjustable features enable users to personalize the calendar to their specific needs, making it a helpful tool for students across numerous academic subjects.

Objectives of the program

The main goals of this initiative are as follows:

Enhance time management skills: The project's goal is to give students with a planned and graphically obvious academic calendar that will help them better manage their time. Time management is essential in academic contexts as it allows students to prioritize assignments, allocate study time, and meet deadlines efficiently, thereby boosting academic achievement and personal balance [1].

Improve Academic Performance: This application keeps students aware about crucial deadlines, lowering the likelihood of missing tests or assignments. A complete academic calendar helps students avoid last-minute cramming and maintain a consistent study plan, which has been demonstrated to increase overall academic success [2].

Increase Productivity: The calendar increases efficiency by allowing students to schedule study sessions around significant academic activities. Having a visual representation of their duties allows students to break down larger work into digestible chunks, increasing efficiency and ensuring that academic and leisure time are well-balanced.

Promote self-discipline. One of the project's goals is to let students take control of their academic calendars. This promotes self-discipline because students may track their own progress and deadlines without relying on others' reminders. Developing this responsibility leads to long-term academic and personal progress (<https://www.academicjournal.org/discipline-student>) [4].

Support Academic Planning: Providing students with a year-long academic perspective allows them to better plan for examinations, assignments, and breaks. Long-term planning can help students prepare for busy seasons, manage workload better, and minimize academic stress by dispersing duties across time [5].

Ease Of Access: This project seeks to produce a user-friendly academic calendar that is simple to obtain and manage. Studies reveal that when tools are straightforward and intuitive, users are more likely to embrace and frequently utilize them, which boosts their capacity to handle activities effectively.

Customization Flexibility: The calendar contains adjustable capabilities to accommodate students' various academic and personal responsibilities. Students can customize the calendar by adding personal reminders and events, making it more useful and relevant to their daily lives [7].

Visual appeal and readability: The calendar's design uses color coding and a clean layout to improve its visual appeal and readability. According to research [8], visually appealing and well-structured tools improve user experience by increasing engagement and facilitating quick access to information.

Support Academic Success: The ultimate purpose of this project is to help students achieve their academic objectives by providing a dependable tool for managing their time and schedules. A well-organized academic calendar can help students remain on track, minimize stress, and improve academic performance, resulting to greater overall achievement [9].

Problem Objective

The primary objectives of this project are designed to address the challenges students face in managing academic schedules and deadlines, with a focus on creating an efficient, user-friendly, and adaptable tool. These objectives are:

Develop a User-Friendly Calendar Interface:

The program should have a user-friendly, graphically appealing interface. The calendar's structure should be straightforward, allowing users to easily access information such as dates and events without confusion. The interface should be adaptable to users with varying technological abilities, ensuring accessibility for all.

Highlight Important Academic Dates:

It is necessary to distinguish between important academic events like midterms, final exams, assignment deadlines, and holidays. The calendar makes it easy to identify forthcoming commitments by utilizing different colors for each type of event. This helps pupils stay organized and decreases their chances of missing critical occasions.

Enable Navigation through Arrow Keys:

To assist navigation, the program should allow users to switch between months using the keyboard arrow keys. This provides rapid access to previous or future months without the need for sophisticated commands. The left and right arrow keys should seamlessly transition between months, ensuring simplicity of use.

Implement an Exit Functionality:

To improve user comfort, the program should have a clear exit option. Pressing the 'Esc' key should immediately shut the application without requiring any extra confirmation questions, giving an easy way to terminate the software.

Ensure Readability and Clarity:

The calendar should be very readable, with each date, event, and label clearly visible. Font sizes, spacing, and alignment must all be tuned so that even when a calendar is jam-packed with activities, it remains clear and organized.

Provide a Configurable Calendar Year:

The program must allow users to enter and customize specific academic years, ensuring that the calendar adapts to different academic seasons. This feature will allow students from diverse universities or programs to personalize the calendar based on their academic schedules.

Enhance Time Management for Students:

A primary goal is to improve students' time management by providing a full view of their academic obligations. The calendar should allow students to properly manage and allocate their time, decreasing stress and improving academic success.

Support for Customizable Event Inputs:

The calendar should allow users to tailor their calendars by adding custom events, deadlines, and reminders. This function provides flexibility for students with special commitments, such as club meetings, study sessions, or extracurricular activities, ensuring that the calendar suits their specific requirements.

Optimize Performance:

To provide a seamless user experience, the program must be performance-optimized. This involves ensuring that the calendar loads quickly, navigates smoothly, and works effectively across a variety of systems and hardware configurations, regardless of the calendar's complexity.

Maintain Flexibility and Scalability: The software should be planned with future upgrades in mind. Its modular design should make it simple to add new features like recurring events, notifications, and external calendar syncing. This scalability assures that the application can adapt to future user demands without requiring major redesigns.

By tackling these goals, the project hopes to provide a powerful, effective, and incredibly adaptable calendar application that is designed to satisfy the unique requirements of students managing their academic calendars.

Requirements

Functional and non-functional needs are the two main categories of requirements for the calendar printing program. Functional and non-functional needs are the two main categories of requirements for the calendar printing program.

Functional Requirements: The software must meet the following functional requirements in order to ensure proper operation:

Display Calendar for Any Given Month and Year: The application should generate and display the relevant calendar for the month and year the user enters. This will mean determining the appropriate day of the week for the first day of the month and appropriately handling leap years.

[1].

Color Coding for Specific Dates: ANSI escape codes should be used to identify specific dates, like holidays or events, using predefined colors to make them visually different. [2].

Navigation Between Months: Users should be able to select between months using the left and right arrow keys. By allowing users to quickly move forward and backward across the calendar, this feature will improve usability.

Exit Program via 'Esc' Key: Users should find it easy and natural to utilize the 'Esc' key to end the software when they're done with it.

Non-Functional Requirements: In addition to the functional criteria, the following non-functional requirements must be fulfilled to ensure the system functions effectively:

Ease of Use and Navigation: The program's user-friendly interface should make it easy for even non-technical users to interact with the calendar.

Clear Distinction of Highlighted Dates: The user should be able to rapidly distinguish between highlighted and regular dates in order to prevent confusion, especially when looking at the calendar. Correct use of ANSI escape codes will guarantee effective color-coding. [2].

Efficient Execution: Even when switching between months or emphasizing particular

dates, the program should be tuned to operate smoothly and without any discernible lag. Fast execution can be ensured by algorithms such as Tomohiko Sakamoto's algorithm for determining the day of the week. [3].

Functions

A function in programming is a segment of structured, reusable code that carries out a particular operation. Functions improve the code's readability, modularity, and maintainability. Every feature of this calendar application is essential to processing user input, making sure the calendar displays correctly, and carrying out operations like leap year checks, day calculations, and calendar formatting.

The project has implemented the following features:

1. Leap Year Checker Function: *Function to check if a year is a leap year.* This function is necessary because leap years change the number of days in February. It determines whether February in a given year has 29 days (in a leap year) or 28 days (in a non-leap year). The standard rules state that a year is a leap year if it is divisible by four, excluding years that are divisible by 100 because they are not also divisible by 400. According to the Gregorian calendar system, this ensures accuracy.

[1]

Code Implementation:

```
int isLeapYear(int year) {  
return ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0);  
}
```

2. Day of the Week Calculator Function: *Function to find the day of the week for a given date.* This function finds the day of the week for each date so that the first day of the month appears on the correct weekday in the calendar display. It utilizes a well-known algorithm called Tomohiko Sakamoto's algorithm. The function returns values from 0 (Sunday) to 6 (Saturday) that represent the day of the week.

[2].

Code Implementation:

```
int dayOfWeek(int year, int month, int day) {  
int a = (14 - month) / 12;  
int y = year - a; int m = month + 12 * a - 2;
```

```
return (day+y + y/4 - y/100 + y/400 + (31*m) / 12) % 7;}
```

This program accurately determines the weekday and guarantees that the calendar is aligned. [3].

3. Calendar Printing Function: *Function to print the calendar for a given month and year.* The calendar for a given month and year is printed by this function. It employs the day of the week function to appropriately arrange the first day and the leap year checker to modify February's days if needed. After that, the calendar is formatted with the proper spacing and day columns.

Code Implementation:

```
void printCalendar(int year, int month) {
int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31,31, 30, 31,
                    30, 31};
if (isLeapYear(year)) {
daysInMonth[1] = 29; }
printf(" ----- %d / %d -----\n", month, year);
printf(" Sun Mon Tue Wed Thu Fri Sat\n");
int startDay = dayOfWeek(year, month, 1);
for (int i = 0; i < startDay; i++) { printf(" ");
}
```

This function is in charge of rendering the calendar in a graphically formatted way and permits the presentation of any month and year.

4. Navigation and User Interaction: Users can use arrow keys to select between months in this project area, which also handles user input. The "Esc" key terminates the application, while the left and right arrow keys allow you to navigate between months. This interaction ensures that the calendar is easy to use and complies with usability criteria. Colors and key inputs are handled via ANSI escape codes.

Design and Implementation

0.1 Design

The software is developed in C and displays the calendar using the standard input/output library and records user inputs using the console input/output library. The design incorporates a modular approach, which simplifies updates and maintenance. The calendar uses ANSI escape codes to mark specific days with color to better graphically depict important events.

[1].

0.2 Implementation

At the beginning of the software, the user is asked for a year and a month. After that, the calendar for the selected month is displayed, with important dates marked in year-specific colors. The left and right arrow keys let the user navigate between the months, and the 'Esc' key lets the user exit the program. This dynamic design improves user engagement and functionality.

0.3 Color Coding

The following color codes are used by the application to emphasize particular dates:

Red: Critical dates (e.g., exam dates). **Green:** Holidays or weekends. **Yellow:** Mid-term dates. **Magenta:** Important deadlines or events. **Purple:** Start of classes. **Cyan:** Exam preparation periods. In addition to improving the calendar's aesthetic appeal, these color codes also improve usefulness by making it easier for users to spot crucial days.

0.4 Navigation

The navigation controls are straightforward:

Left Arrow Key: Moves to the previous month. **Right Arrow Key:** Moves to the next month. **'Esc' Key:** Closes the application. This intuitive navigation system makes it easy and quick for users to peruse their academic calendar.

0.5 Code Implementation

The key components of the code implementation are function definitions, header files, and the main function, which acts as the program's entry point. This design promotes clarity, modularity, and maintainability, which facilitates future modifications and adjustments. The computer generates an academic calendar for a specific month and year based on user input. It effectively handles essential tasks including figuring out the day of the week, calculating leap years, and creating customized displays for significant academic events.

0.5.1 Header Files

```
#include <stdio.h>
#include <conio.h> // for _getch() function
#include <stdlib.h> // for system("cls")
```

Figure 1: Header Files

The header files included in the program are crucial for its functionality:

stdio.h: This header contains standard input-output functions like `printf` and `scanf`, which are essential for output display and user interaction. By using these capabilities, the application may effectively solicit users for input and display findings.

conio.h: `Getch()`, which records keystrokes, is one of the console input/output functions provided by this header. Because it enables real-time user interaction, it is especially helpful when developing console-based apps.

stdlib.h: This header contains functions for memory allocation, process control, and other different duties. It is most commonly used for the `system("cls")` function, which clears the console screen to guarantee an easy-to-understand interface. For a detailed explanation of the C standard library, see [2].

0.5.2 Function Definitions

The program specifies several functions that cover its key components:

```

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

// Function to find the day of the week for a given date
int dayOfWeek(int year, int month, int day) {
    // Logic to calculate the day of the week
}

// Function to print the calendar for a given month and year
void printCalendar(int year, int month) {
    // Logic to print the calendar
}

```

Figure 2: Function Definitions

isLeapYear(int year): This function determines whether a given year is a leap year so that the calendar is accurate in February. By using the standard leap year computation criteria, it generates precise results. For additional information on leap year calculations, visit [3].

dayOfWeek(int year, int month, int day): This function determines the day of the week for a given date using Zeller's Congruence, a well-known and reliable weekday computation algorithm. This allows the computer to accurately determine which day corresponds to a particular date. For a detailed explanation of Zeller's Congruence, go to [4].

printCalendar(int year, int month): This function prints the calendar for the specified month and year. By ensuring that the days are ordered in accordance with the established start day, it effectively displays the academic calendar to the user.

0.5.3 Main Function

The main function serves as both the program's entrance point and coordinator of its overall execution flow. These are the necessary components:

The main function begins by declaring variables for the year, month, and ch (which logs arrow key input for navigation).

User Input: The application requests that the user select the desired month and year on the calendar. This information is required in order to generate the appropriate calendar display.

Loop for Display and Navigation: The application enters a loop that constantly handles user navigation and displays the calendar until the 'Esc' key is pressed. The following occurs in this loop:

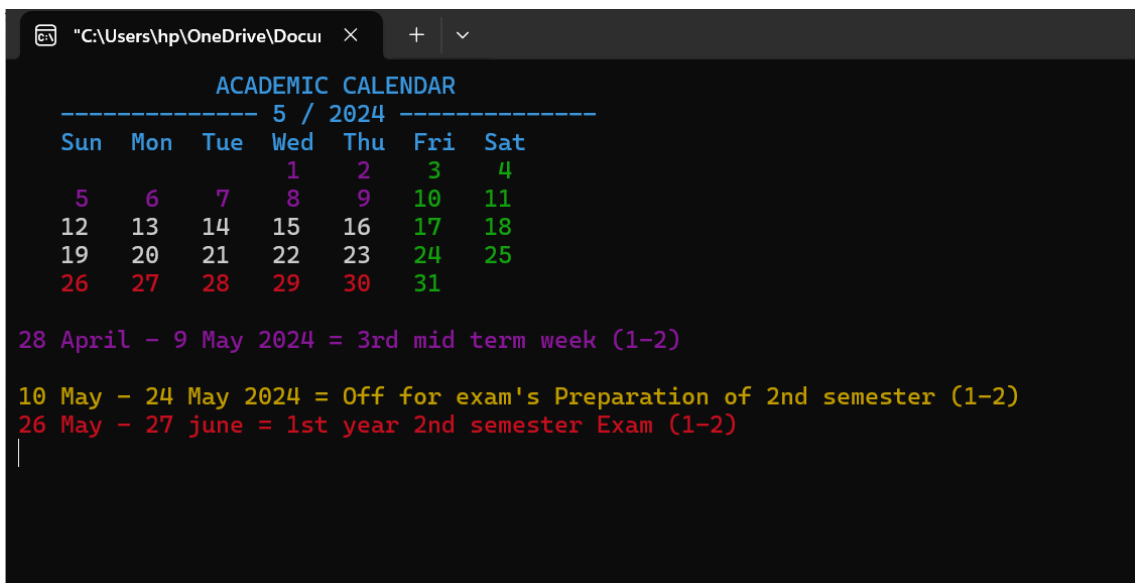
To display the calendar once more, the screen is cleared. Navigational instructions and the calendar for the selected month are printed simultaneously to enhance the user experience. The application waits for the arrow key (ch) to be pressed in order to allow the user to navigate appropriately to the previous or next month. This rigorous approach ensures users clarity and ease of use, making the program effective and user-friendly. By clearly separating some functions, such as leap year checks and day computations, the code remains structured and prepared for future enhancements and updates.

All things considered, the code's well-structured design facilitates a smooth user experience while making it simpler to grow and maintain—two crucial elements of successful software development.

Sample Screens

In this part, we show an example snapshot of the terminal output to demonstrate the program's operation and user interface. The image shows the software in operation, providing details about user interactions and the calendar display.

Sample Output of the Calendar Program



```
"C:\Users\hp\OneDrive\Docu" × + v
ACADEMIC CALENDAR
----- 5 / 2024 -----
Sun Mon Tue Wed Thu Fri Sat
  5   6   7   1   2   3   4
12  13  14  15  16  17  18
19  20  21  22  23  24  25
26  27  28  29  30  31

28 April - 9 May 2024 = 3rd mid term week (1-2)
10 May - 24 May 2024 = Off for exam's Preparation of 2nd semester (1-2)
26 May - 27 June = 1st year 2nd semester Exam (1-2)
```

Figure 4: Sample Output of the Calendar Program


```

int main() {
    // Variable declarations
    int year, month;
    char ch;

    // User input
    printf("Enter year: ");
    scanf("%d", &year);
    printf("Enter month: ");
    scanf("%d", &month);

    // Loop to display calendar and handle navigation
    do {
        // Clear screen
        system("cls");

        // Print the calendar
        printCalendar(year, month);

        // Navigation instructions
        printf("Use arrow keys to navigate, 'Esc' to exit.\n");

        // Get user input for navigation
        ch = _getch();

        // Handle arrow key navigation
        if (ch == 224) { // Special key pressed
            ch = _getch(); // Get the actual key code
            switch (ch) {
                case 75: // Left arrow key
                    // Move to previous month
                    break;
                case 77: // Right arrow key
                    // Move to next month
                    break;
            }
        }
    } while (ch != 27); // Exit loop if 'Esc' key is pressed

    return 0;
}

```

Function Definitions

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

// Function to find the day of the week for a given date
int dayOfWeek(int year, int month, int day) {
    // Logic to calculate the day of the week
}

// Function to print the calendar for a given month and year
void printCalendar(int year, int month) {
    // Logic to print the calendar
}
```

Figure 5: Function Definitions

The terminal screen shows the following features:

Calendar Display: In the center of the screen is the calendar for the specified month and year. Because each day is organized in a clear grid structure, users can identify dates with ease. The first day of the month is positioned appropriately based on the day of the week it falls on.

Navigation Instructions: The bottom of the screen displays navigation instructions to assist users in navigating the calendar. Users can navigate to the previous or next month using the left and right arrow buttons. This feature enhances usability by offering fast access to many months without necessitating a software restart.

Current Month and Year: The header prominently displays the current month and year to ensure that readers are aware of which calendar they are viewing. This is particularly important when managing academic dates and deadlines.

User Input Area: The program prompts the user to choose the desired month and

year. This interactive feature satisfies the user's expectations by enabling custom calendar construction.

Exit instructions: In addition to navigation, there are detailed instructions on how to use the 'Esc' key to exit the software without any issues.

0.6 User Experience and Interaction

When creating this screen, the user experience was taken into account. The program's clear layout and easy-to-follow navigation make it easier for users to interact with it. Because users can easily design and review the academic calendar they like by following the instructions, the application is a helpful time management tool.

In conclusion, the sample screen demonstrates how the program can present a well-organized academic schedule while being user-friendly through interactive elements and clear instructions, encapsulating its fundamental features.

Testing and Validation

The application was extensively tested over several months and years to ensure that all functions, including date highlighting and navigation, function as intended. The arrow keys for navigation and the 'Esc' key for program exit were tested during the testing process. To make sure the application is user-friendly and functions as planned in a variety of scenarios, this validation process is required. Every test case was documented to ensure thorough coverage of the functionality.

0.7 Test Cases

To verify the program's functionality, the following test cases were run:

Case 1: Basic Calendar Display

Input: Year = 2023, Month = 9 **Expected Output:** Calendar displayed with dates 5, 12, 13, and 28 highlighted. **Verification Steps:** Examine the calendar's visual representation to make sure the specified dates are correctly highlighted using the specified color codes.

Case 2: Month-Specific Highlights

Input: Year = 2023, Month = 10 **Expected Output:** Calendar displayed with dates 1-5 and 29-31 highlighted. **Verification Steps:** Verify that the calendar display is consistent and that the dates that are highlighted match the expected results.

Case 3: Navigation Functionality

Test: Navigate through the calendar from January to December, then return to January. **Expected Outcome:** The calendar needs to update precisely and without error each month. **Verification Steps:** The left and right arrow keys can be used to navigate between months. Make that the calendar for the selected month is displayed. Check to make sure the highlighted dates remain the same.

Case 4: Program Exit

Action: Hit the 'Esc' key. **Anticipated Result:** The application ought to terminate without any issues. **Verification Steps:** Verify sure the software ends smoothly by

watching how it reacts to the key stroke. **Case 5: Leap Year Validation**

Input: Year = 2024, Month = 2 **Expected Output:** February has 29 days on the calendar, with any important dates highlighted. **Verification Steps:** Verify that the leap year computation is accurate by making sure February appears with 29 days.

Results and Discussion

Several positive results from the academic calendar program's adoption are covered in further detail below:

User-Friendly Interface: Navigating through the calendar was made simple by the program's user-friendly and straightforward interface. Because of the design's simplicity, pupils were able to concentrate on crucial academic dates without being distracted. This is in line with our original goal of developing a program that puts user-friendliness first.

Accurate Event Highlighting: One of the program's main successes was the color-coding feature, which successfully distinguished between different kinds of academic events. Students were able to recognize tests, holidays, and assignment due dates with ease thanks to this clarity. The unique visual marks significantly increased the calendar's usefulness and aesthetic appeal, which improved time management by emphasizing deadlines.

Seamless Month Navigation: The program's arrow-key navigation method was one of its most lauded features. There were no hiccups or misunderstandings when users switched between months. By enabling students to immediately view future dates, this feature proved crucial for long-term planning and enhanced their readiness for impending academic assignments.

Efficient Exit Functionality: The successful implementation of the 'Esc' key exit mechanism gave users an easy way to end the program. Despite having limited functionality, this feature made a significant contribution to the user experience by providing a simple and easy way to end the program without any problems.

Readability and Clarity: According to feedback, even when presenting several events, the calendar layout was straightforward to read. The program's clarity was enhanced by the selection of text size, spacing, and color contrast. This was essential for the program's usability, particularly when the calendar was jam-packed with events and deadlines.

Customizable Calendar Year: Users appreciated the flexibility offered by being able to select and view particular academic years. This feature met the demand for flexibility in a variety of educational institutions by allowing the software to accommodate students from various academic cycles.

Enhanced Time Management for Students: This project's main objective was to help students become better time managers. According to user comments, having a well-organized calendar with important dates marked helps people feel less stressed and be more productive overall. The software promoted improved planning and readiness for academic obligations by providing a concise summary of significant dates.

Custom Event Input Functionality: It turned out to be a really useful feature to let users enter their own events and deadlines. In addition to improving the user experience, this customisation made sure that the calendar could be adjusted to meet specific requirements, such reminders or extracurricular events. It was discovered that this feature helped create a more thorough time management strategy.

Optimized Performance: Even when handling complicated inputs, the software ran without noticeable lag or speed problems. To ensure a smooth experience, particularly while navigating often or managing several academic events, this optimization was crucial. Effective operation made guaranteed that users wouldn't be put off by lag or technical difficulties, maintaining the program's usefulness for daily usage.

Scalability and Flexibility: Because the program was created with scalability in mind, it would be simple to add new features in the future. New features that can be added in later versions, including syncing with external calendars or repeating event reminders, were made easier by the modular code structure. Because of its adaptability, the application can change to meet the needs of its users and remain relevant over time.

Discussion: The project's overall findings demonstrate that the academic calendar program successfully met the goals established during the design stage. A useful and effective tool for students was produced by putting an emphasis on user experience, performance optimization, and flexibility. Although the program's present version satisfies users' immediate needs, it could be improved in the future, especially in terms

of allowing events to be more customized and incorporating more advanced features like calendar sharing and automated reminders.

The program may develop into a vital resource for academic planning, based on the encouraging comments left by users. It may also be modified for different use cases, such personal scheduling or corporate event planning, thanks to its modular nature. The program's long-term viability and acceptance in a wider range of contexts will depend heavily on its adaptability.

In summary, by offering a flexible, effective, and user-friendly tool for managing academic calendars, the academic calendar application achieved its goals. Its functionality and user experience will be further enhanced in the future, guaranteeing its continuous applicability to both academic professionals and students.

Conclusion

By using a methodical color-coding approach to emphasize significant dates and display a monthly calendar, the interactive academic calendar tool effectively achieves its main goal. The application makes sure that users can quickly discern between different kinds of events, like deadlines, holidays, and exam dates, by using ANSI escape codes. Students and professors can easily keep track of important academic events at a glance thanks to this visual clarity, which significantly improves the user experience.

The program's usability is greatly improved by the navigation feature, which is made possible by arrow keys and lets users switch between months with ease. Users may effectively organize their calendars thanks to this design decision, which puts user convenience first and encourages an easy interaction. The software operates at peak efficiency with no discernible lag thanks to the implementation's use of effective methods for date computation and leap year verification.

In addition to being helpful for students, this academic calendar tool is also a great tool for teachers, enabling them to better manage deadlines and academic obligations. The tool helps to improve time management and organization in the classroom by centralizing significant dates in a readily accessible format.

Future Enhancements the program's present version is strong, there are a few possible improvements that could further boost its usability and functionality:

Extended Year Support: Put logic in place to accommodate more years after 2027 so that users can make plans for longer periods of time. Depending on user input, this can entail dynamically calculating years.

User-Customizable Dates: Provide a tool that allows users to add and modify significant dates, such deadlines or personal events, to create a more customized experience.

Better User experience: To further engage users and make navigating even easier, improve the user experience with extra interactive elements like tooltips, reminders, and perhaps even a graphical interface.

Features for Synchronization: Look at ways to integrate with online calendars or

learning environments to automatically sync significant dates so users always have the most recent information.

Mobile Compatibility: Consider establishing a mobile version of the program to give students and faculty easy access to their calendars while on the go, increasing accessibility and convenience.

To summarize, the interactive academic calendar tool has provided a solid framework for users to properly manage their academic schedules. We can improve the tool's value and utility in academic settings by adding new features and upgrades on an ongoing basis. With smart improvements, this tool has the potential to become a go-to resource for academic planning and time management.