CS-2303, System Programming Concepts, C-term 2017

Project 1 (20 points)
Assigned: Thursday, January 12, 2017
Due: Saturday, January 21, 2017, 6:00 PM

Programming Assignment #1 — Display a 12-month Calendar

Abstract

Write a *C* program called **`PA1.c`** that displays a twelve month calendar for an arbitrary year. Prompt the user for the year of the calendar, and print out the calendar month by month, so that it looks like a real calendar.

Outcomes

After successfully completing this assignment, you should be able to:–

- Develop a *C* program on a Unix or Linux platform
- Design a program that contains nested selection and iteration constructs and separate functions
- Specify the loop invariants that you use to reason about your program
- Use advanced formatting strings and conversion specifiers to do I/O in a *C* program

Before Starting

Re-read Chapter 1, Chapter 2, and sections §§3.1-3.5 K&R. These chapters should be very easy because of the similarity to Java. It is also suggested that you complete *Lab #1*, either during the scheduled lab session or during your own time.

The Assignment

Write a *C* program that displays twelve-month calendar for a particular year. The program should prompt the user for the year to be printed, and then it should figure out (a) whether the year is a leap-year and (b) what day of the week the chosen year starts on.

The calendar should be formatted as shown in the sample execution below. Note that numbers the days must be right-justified under the names of the days and that two spaces separate the names of the days from each other.

Interfaces

The interface **`<stdio.h>`** provides the functions **`printf`** and **`scanf`**.

Assumptions and Restrictions

The user may enter any positive integer for the year. You must calculate the calendar according to the modern international standard calendar (introduced by Pope Gregory XIII in the year 1582). For input years earlier than 1582, calculate them as if the modern calendar were in effect. In the modern calendar, years that are divisible by 4 are leap years, except that years divisible by 100 are not leap years unless they are also divisible by 400. That is, there are 97 leap years every four centuries.

You will have to figure out which day of the week the calendar starts on. You may do this by referring to a known year in which you know the day of the week of a particular date. You will then work backwards from that known date to find the start of the input year.[1]

Sample Execution

```
                          MONTHLY CALENDAR

Please enter year for this calendar:- 2009


  ***    CALENDAR for 2009   ***

January 2009

Sun  Mon  Tue  Wed  Thu  Fri  Sat
                       1    2    3
  4    5    6    7    8    9   10
 11   12   13   14   15   16   17
 18   19   20   21   22   23   24
 25   26   27   28   29   30   31

February 2009

Sun  Mon  Tue  Wed  Thu  Fri  Sat
  1    2    3    4    5    6    7
  8    9   10   11   12   13   14
                 .
                 .
                 .
```

(Output continues for all 12 months. Note that dates must be *right justified* under day names.)

Implementation Notes

Since we have not yet studied arrays, strings, or arrays of strings, you should design your algorithm to use **`if-else`** or **`switch`** statements to print the month names and to set other variables.

You should partition your program into at least three functions. Here is an example partition:–

- The function **`main()`** prompts the user for input, calls a function of your own design to determine the starting day of the input year. It then invokes the function **`printCalendar()`** to actually print the twelve month calendar.
- The function **`printCalendar()`** takes two arguments, the year number and the starting day. It then loops through the year and calls the function **`printMonth()`** twelve times, once for each month.
- The function **`printMonth()`** takes three arguments, the year number, the month number and the starting day of that particular month, and it returns the number of the day on which the next month starts. Print month has to first call a function **`printMonthName()`** and then print out the days of the month in calendar format.

---

[1] Little known fact:– The number of days in four centuries is exactly divisible by seven. This means that each four-century interval starts on the same day of the week.

- The function **`printMonthName()`** takes the year number and the month number as arguments, prints out the line identifying the month, and returns the number of days in that month, taking into account leap year. The example output of **`printMonthName()`** should look resemble the following:–

  ```
  January 2009
  ```

  Since we are not using arrays of strings, **`printMonthName()`** should use a **`switch`** statement to select and print the name of the month and to determine the number of days in that month. If the month is February, it should also figure out whether the year is a leap year and return the correct number of days.

Algorithm and Loop Invariants

There are many sources on the web and at WPI for a suitable algorithm for this assignment. You may consult any of these, *but you must cite your source*. If you worked out the algorithm on your own, you should say in your write up file that this is entirely your work.

Note: If you borrow some or all of an algorithm from someone else or from somewhere else, *do not copy it*. Write it out in your own words and your own coding style. Also, please explain enough about how the algorithm works that the graders can conclude that you understand it.

This project requires at least two loops. For each loop, write a *loop invariant* — that is, a logical statement in English or mathematical notation that says what salient facts are true about the relationships of the variables at the same point in the loop for each iteration.

Write each loop invariant as a comment in the body of the loop at the exact point during the execution of the loop body where the invariant is **`TRUE`**. Also include copy each loop invariant into your **`README`** document below.

Deliverables

Write a document called **`README.txt`**, **`README.doc`**, or **`README.docx`** summarizing your program, how to run it, and detailing any problems that you had. Also, if you borrowed all or part of the algorithm for this assignment, be sure to cite your sources *and* explain in detail how it works. Be sure also to describe the *loop invariant* of each loop.

From browser, submit *C* source code file and **`README`** to *Canvas*.

Note: This course is listed as two separate lecture "courses" plus one common "Labs course." You are registered for one of the Lecture courses and also for the Labs course. This, along with all other programming assignments must be submitted to the *Assignments* section of the "Labs" course.

Programs submitted after the due date (Saturday, January 21, 2017, 6:00 PM) will be tagged as late, and will be subject to the late homework policy.

Grading

Graders will compile your assignment by executing the following command on an *Ubuntu* Linux system compatible with your course virtual machine:–

```
gcc -Wall-o PA1 PA1.c
```

Your program must *compile without errors in order to receive* any *credit* for this assignment. You *must not* use any extra switches — for example, **`-ansi`** or **`-std=C99`**. These will cause incompatibilities that will cause it to fail to compile for the graders. If you do your work on some other system, you may find

that your system adheres to a slightly different standard and that some details of the *C* language may be different from those on the course virtual machine. Before submitting your assignment, be sure that it compiles on the course virtual and correct it if it does not.

This assignment is worth twenty (20) points:–

- Correct compilation without warnings – 2 points
- Correct execution with graders' test cases – 2 points
- Correct usage of **`scanf()`** to get inputs from user – 1 point
- Correct usage of **`print()`** to print the various lines of the calendar – 3 points
- Correct usage of conditional and loop statements – 5 points
- Satisfactory **`README`** file – 2 points
- Loop invariant for each loop in comments in the code *and also* in the **`README`** document – 5 points