Assignment 2

COMP 206, Fall 2016
Due: Friday October 21st, (23:59) via My Courses
100 marks total

Warm-Up Exercises

Not for marks. The TAs or instructor will solve these for you in office hours, if you wish.

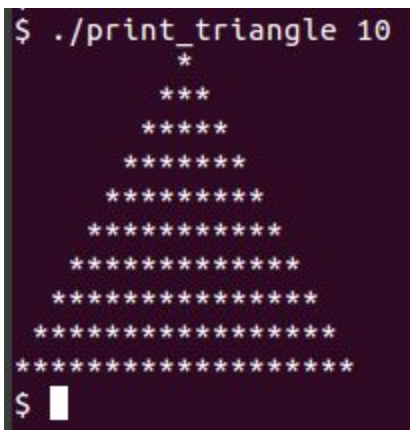
Tutorial 1 - Sorting Integers

A text file contains between 1 and 100 positive integers, all with values between 0 and 32000, written one per line (tut1_input.txt is a sample file). Write a program that:

- Takes the filename as its only command-line argument
- Reads the integers from the file
- Prints them to standard output SORTED from smallest to largest

Tutorial 2 - Triangle Printing

AASCI art refers to the creation of interesting patterns using only white-spaces and various characters that can be printed to the terminal. Write a program that prints a triangle that is correctly centred on the line with height (in number of rows) specified as the only command-line argument. Example:



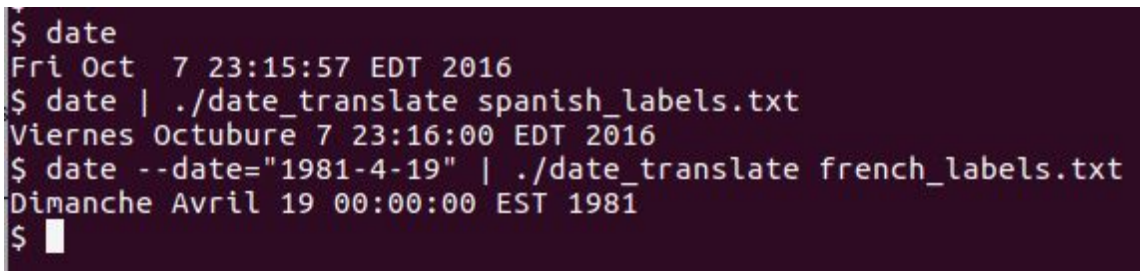
1. Translate the Date (35 marks)

An important part of software systems is to ensure that they can function equally well in any of the world's many languages. One method to allow this is to read "localization" data from files, rather than hard-coding directly into source code. We can define a localization file format needed for translating dates as a 2-line text files that contains the names of all 7 days of the week on one line (space separated) and the 12 months of the year on a second line.

Now we can sort of change languages (we must ignore forming proper sentences for now) just by pointing our program to a different file. 3 samples are provided as: english_labels.txt, french_labels.txt, and spanish_labels.txt.

Write a program that translates dates by:

- Accepting a date on standard input (can be typed on the terminal, but more to the point, will be piped to you from "date").
- Accepts one and only command-line argument: the path to a language file with the format described above
- Outputs a date with the same format as the input, but with both the day name and the month name translated as requested (output the full day and month names, not date's typical 3 letter version).



(Note the --date=... option of the date command lets you easily test different months and days quickly without having to wait overnight)

2. Multi-Language Calendar (65 marks)

Write a program that prints one year's full calendar to standard output following a precise output format. The language of the calendar will be configurable based on the same input text format as Question 1. Your program will accept exactly 3 command-line arguments:

- Language file as the path to a text file as was described above
- Maximum number of characters **DAYSIZE** to print for the day-of-week labels, which must be 2 or greater
- The day-of-week that starts the year as an integer from 1 to 7 (1 is Sunday in English)

Your output must ***precisely*** follow these examples (text specification to follow):

```
$
$ ./calendar
Usage: ./calendar label_file day_spacing first_day
$ ./calendar english_labels.txt 6 3
************************************************************
* January
************************************************************
* Sunday * Monday * Tuesda * Wednes * Thursd * Friday * Saturd
************************************************************
*        *        * 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     *        *        *
************************************************************
* February
************************************************************
* Sunday * Monday * Tuesda * Wednes * Thursd * Friday * Saturd
************************************************************
*        *        *        *        * 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     *
************************************************************
* March
************************************************************
* Sunday * Monday * Tuesda * Wednes * Thursd * Friday * Saturd
************************************************************
*        *        *        *        *        *        * 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     *        *        *        *        *        *
```

```
$
$ ./calendar french_labels.txt 2 7
************************************
* Janvier
************************************
* Di * Lu * Ma * Me * Je * Ve * Sa
************************************
*    *    *    *    *    *    * 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 *    *    *    *    *    *
************************************
* Fevrier
************************************
* Di * Lu * Ma * Me * Je * Ve * Sa
************************************
*    * 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 *    *    *    *
************************************
* Mars
************************************
* Di * Lu * Ma * Me * Je * Ve * Sa
************************************
*    *    *    * 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 *    *
```

(Note that the image just shows the first 3 months of the calendar to fit on the page, but your output must continue for the full year)

12 months are printed. Each month is composed of:

- A line of star characters that span the calendar's full width (max of any other line specified)
- A line composed on a single star, a single space and the month's name in the requested language
- A line of star characters that span the calendar's full width
- A line with 7 columns worth of day labels, where each day label contains, in order:
    - A single star
    - A single space
    - A day-of-week label cut down to **DAYSIZE** characters if the label is too long
    - Additional spaces to pad up to **DAYSIZE** if the label was too short
    - A single space
- A line of star characters that span the calendar's full width
- 5 or 6 lines that contain the days of that month:
    - Print exactly 30 days always (avoids leap-years etc).
    - Lay out the date numbers with the usual logic:
        - The first day of one month is on the day following the last day of the previous month
        - When a week is not "full" with dates, blank days are printed to maintain spacing, both at the start and end of the month
- Each day section must contain, in order:
    - A single *
    - A single space
    - The date as a one or two digit integer, following correct ordering
    - Additional spaces to pad up to **DAYSIZE** if the number was too short
    - A single space

Submitting your solutions:

You will submit a single zip file that contains the 2 C files holding your solutions. Create the zip file with the command:
$ zip A2_solutions.zip q1.c q2.c

The TAs will mark your code by compiling it, and running several different test cases. It must compile without errors and produce an output program to be awarded any marks, so test carefully at Trottier as always.

For this assignment, we will increase the required level of input checking. We may try to run with files that do not exist, integer values that are outside the proper ranges or negative values, or strings instead of integers, for example. The proper behaviour for bad inputs is for your code to detect the situation, print “ERROR” (more description is useful but not considered in marking), and return immediately, rather than running in some incorrect way.