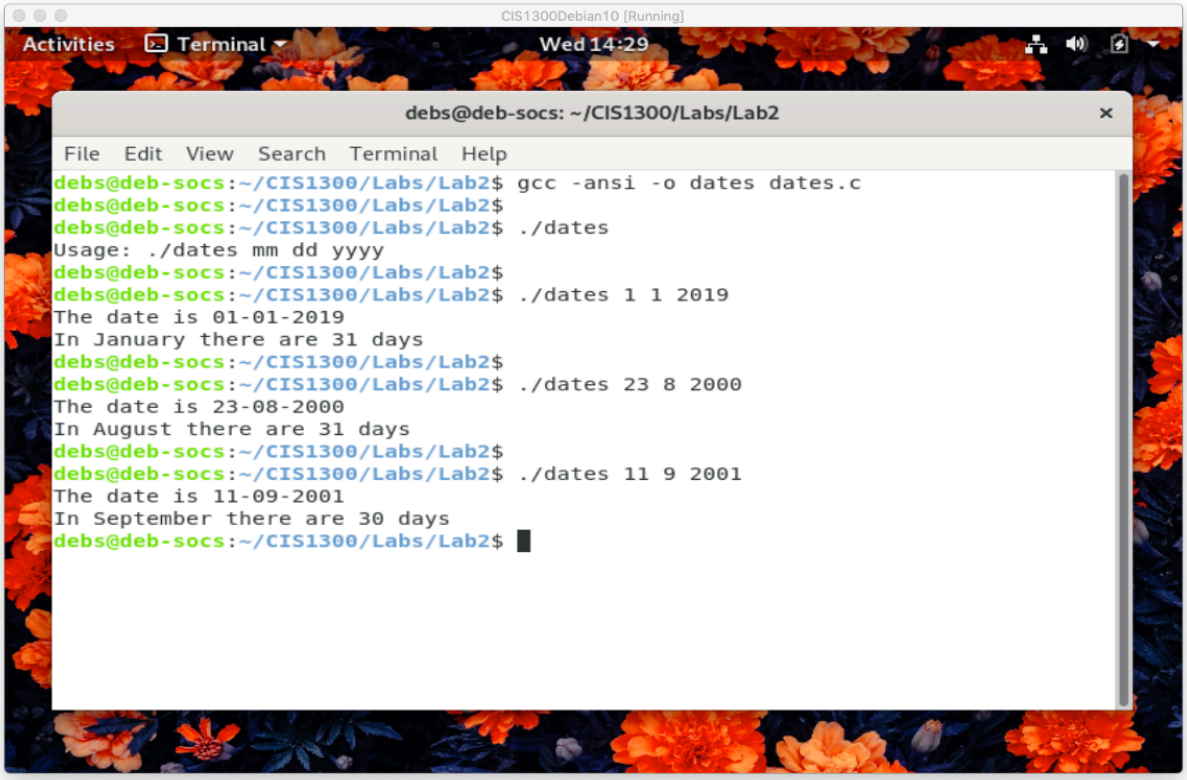
Week 2 - Lab 2 – A Brief Look at Time

In this lab you will take a program and change it to do various related tasks. The program is in the file **`dates.c`** on the CourseLink site in Lab 2. Download this file and open in your favourite editor and you will see:

```
#include <stdio.h>
#include <stdlib.h>

/*
 *   Program name: dates.c
 *   Author: Deb Stacey
 *   Last Update: September 11, 2019
 *   Function: to print out date given on command line
 *   Compilation: gcc -ansi -o dates dates.c
 *   Execution: ./dates 23 8 2019
 */

int main ( int argc, char *argv[] ) {

   /* Names of the months */
   char *monthName[12] = { "January", "February", "March", "April", "May",
"June", "July", "August", "September", "October", "November", "December" };

   /* The number of days in each month */
   int monthLength[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

   int dd = 0;
   int mm = 0;
   int yyyy = 0;

   if ( argc < 4 ) {
      printf ( "Usage: ./dates mm dd yyyy \n" );
      return ( 1 );
   } else {
      dd = atoi ( argv[1] );
      mm = atoi ( argv[2] );
      yyyy = atoi ( argv[3] );
   }

   /* Remember that arrays like monthName and monthLength start their index
      at 0 and not 1. */
   /* The first entry in the array is monthName[0], monthLength[0] */

   printf ("The date is %02d-%02d-%04d\n", dd, mm, yyyy);
   printf ("In %s there are %d days\n", monthName[mm-1], monthLength[mm-1]);

   return ( 0 );
}
```

Compile and run the program to check on its behaviour:



You can see that the program expects to see 3 numbers on the command line that represent the day, month, and year of a date. If you do not enter 3 numbers the program will print out a usage message and stop.

If you give it 3 numbers it will print out this date and tell you how many days are in the month in that date.

In the code you will see two very interesting variables:

```
/* Names of the months */
char *monthName[12] = { "January", "February", "March", "April",
"May", "June", "July", "August", "September", "October", "November",
"December" };

/* The number of days in each month */
int monthLength[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
};
```

These variables are called arrays. An array stores multiples of the same type, e.g. integers. For example `monthLength` is an integer array that has room to store 12 integers. `monthName` stores 12 character strings (this is more complicated and will be explained in another class). The important thing to remember about arrays is how you refer to an individual item within the array. How do I get the first integer in the `monthLength` array? Each item in an array is "indexed" or "has a number" and you put that number in [ ] after the array name, e.g.

`monthLength[10]`. But the tricky part is that the index starts at 0 and not 1! So the first integer in the `monthLength` array is `monthLength[0]`.

After you understand what the **`dates.c`** program is doing, you need to change it to do the following 4 tasks:

1. Instead of printing out "The date is dd-mm-yyyy" change this to "The date is monthName dd, yyyy" and do not print out the next like about the number of days in the month.
   ```
   $ ./dates 2 3 2019
   The date is March 02, 2019
   ```

2. Next, add code to check if the month input on the command line is in the range 1 to 12. If it is not then print out the following error message and exit the program. Make sure that your return code is non-zero.
   ```
   $ ./dates 1 13 2019
   Error - the month entered (13) is not in the proper range (1-12)
   ```

3. Next, add code to check if the day input on the command line is in the range 1 to the number of days in the month. If it is not then print out the following error message and exit the program. Make sure that your return code is non-zero.
   ```
   ./dates 31 4 2019
   Error - you entered 31 for the day and that is not in the range
   (1-30)
   ```

4. Lastly write code to determine if the year (yyyy) is a leap year. The algorithm to check if a year is a leap year is as follows:

   Every year that is exactly divisible by four is a leap year,
   except for years that are exactly divisible by 100
   unless they are exactly divisible by 400.

This lab is worth 2 marks. Each task is work 0.5 of a mark. If you have time you can work on a bonus for 1 mark.

Bonus

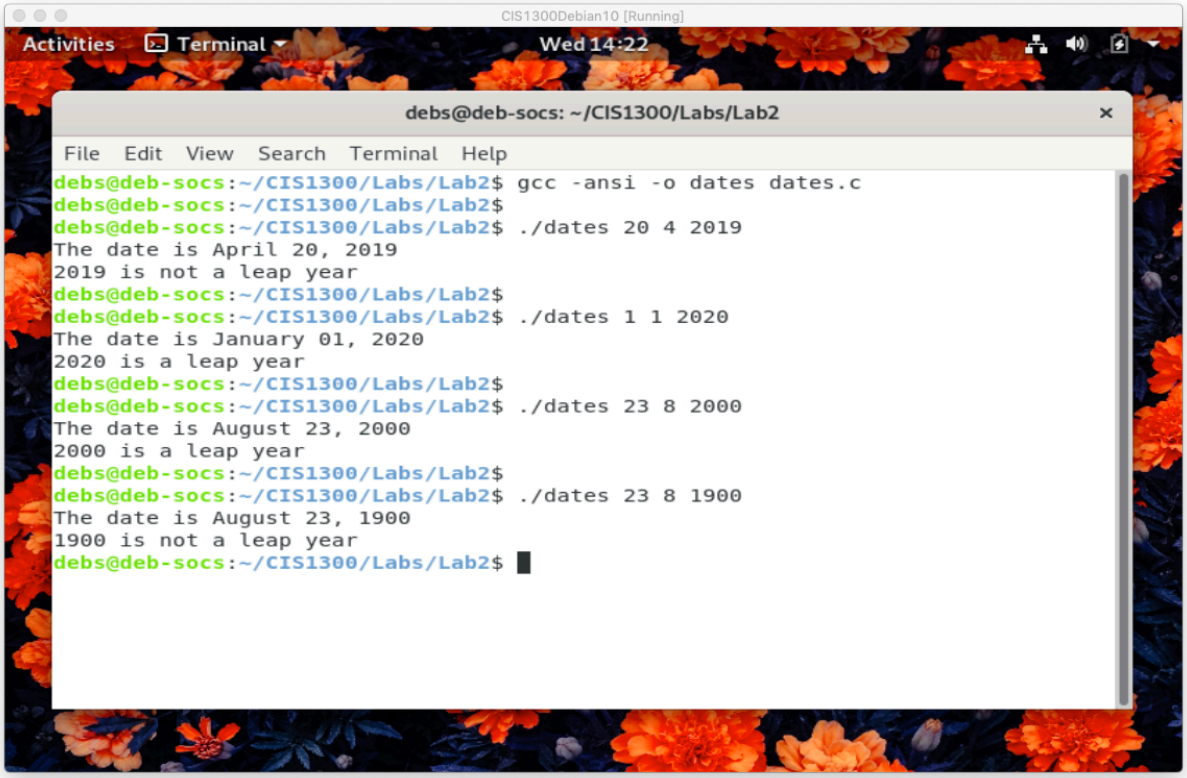
- Rewrite Task 3 so that it takes leap years into account. In other words, February 29 is a proper date if the year is a leap year. So February 29, 2020 (29 2 2020) is proper but February 29, 2019 (29 2 2019) is not.

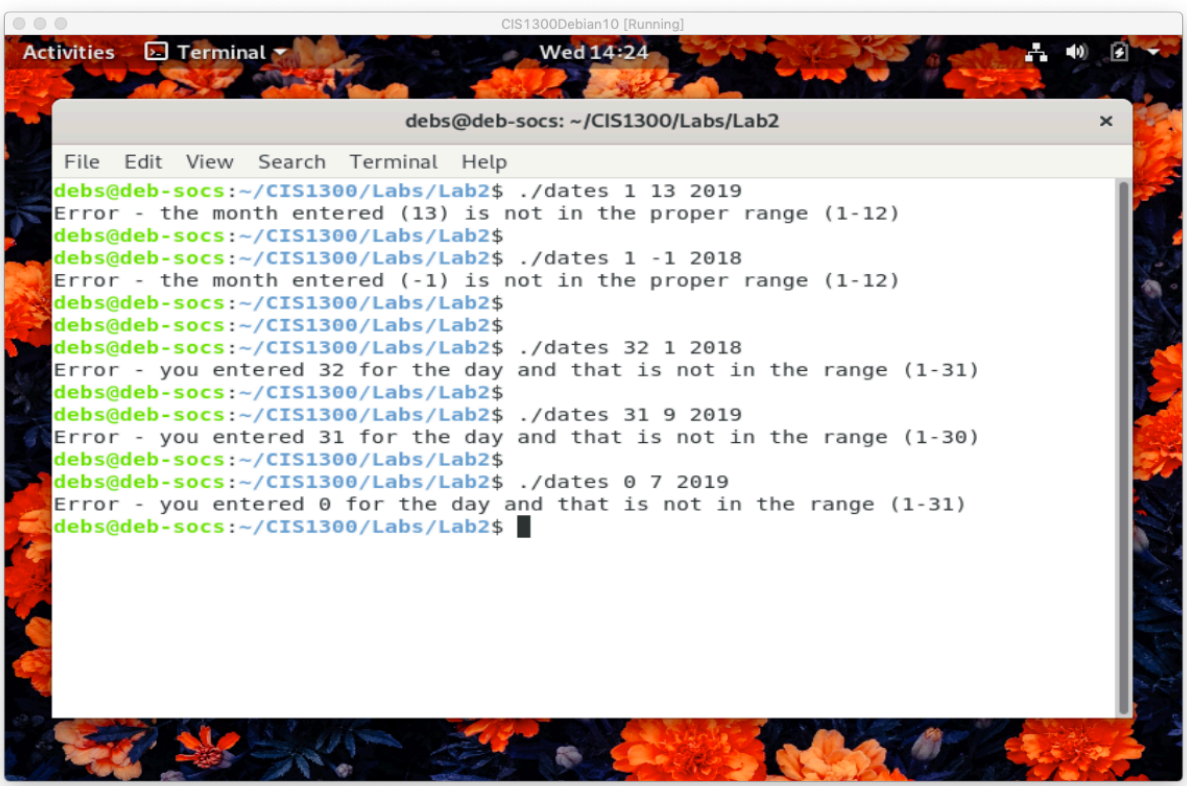
Checking Your Work

Download the file **`checkLab2.sh`** from the Lab 2 site on CourseLink and do the following after you have compiled your **`dates.c`** program. Show the TA the output of this program before you leave the lab so that they can record your grade. If you have done the bonus check it with **`checkL2Bonus.sh`**.

```
$ sh checkLab2.sh
$ sh checkL2Bonus.sh
```

The following are some screen shots that show you how your program should behave:





CIS1300Debian10 [Running]

Activities Terminal Wed 14:26

debs@deb-socs: ~/CIS1300/Labs/Lab2

File Edit View Search Terminal Help

```
debs@deb-socs:~/CIS1300/Labs/Lab2$ ./dates 28 2 2019
The date is February 28, 2019
2019 is not a leap year
debs@deb-socs:~/CIS1300/Labs/Lab2$
debs@deb-socs:~/CIS1300/Labs/Lab2$ ./dates 29 2 2019
Error - you entered 29 for the day and that is not in the range (1-28)
debs@deb-socs:~/CIS1300/Labs/Lab2$
debs@deb-socs:~/CIS1300/Labs/Lab2$ ./dates 28 2 2016
The date is February 28, 2016
2016 is a leap year
debs@deb-socs:~/CIS1300/Labs/Lab2$
debs@deb-socs:~/CIS1300/Labs/Lab2$ ./dates 29 2 2016
The date is February 29, 2016
2016 is a leap year
debs@deb-socs:~/CIS1300/Labs/Lab2$
```

debs@deb-socs: ~/CIS1300/Labs/Lab2

File Edit View Search Terminal Help

```
debs@deb-socs:~/CIS1300/Labs/Lab2$ sh checkLab2.sh
Checking that the program handles correct input...

./dates 20 4 2019
The date is April 20, 2019
2019 is not a leap year

./dates 1 1 2020
The date is January 01, 2020
2020 is a leap year

./dates 23 8 2000
The date is August 23, 2000
2000 is a leap year

./dates 23 8 1900
The date is August 23, 1900
1900 is not a leap year

Checking that the program handles incorrect input...

- no commandline argument
./dates
Usage: ./dates mm dd yyyy

- incorrect months
./dates 1 13 2019
Error - the month entered (13) is not in the proper range (1-12)

./dates 1 -1 2018
Error - the month entered (-1) is not in the proper range (1-12)

- incorrect days
./dates 32 1 2018
Error - you entered 32 for the day and that is not in the range (1-31)

./dates 31 9 2019
Error - you entered 31 for the day and that is not in the range (1-30)

debs@deb-socs:~/CIS1300/Labs/Lab2$
```

debs@deb-socs: ~/CIS1300/Labs/Lab2

File Edit View Search Terminal Help

```
debs@deb-socs:~/CIS1300/Labs/Lab2$ sh checkL2Bonus.sh
Checking Bonus - number of days in Feb adjusted for leap year

./dates 28 2 2019
The date is February 28, 2019
2019 is not a leap year

./dates 29 2 2019
Error - you entered 29 for the day and that is not in the range (1-28)

./dates 28 2 2016
The date is February 28, 2016
2016 is a leap year

./dates 29 2 2016
The date is February 29, 2016
2016 is a leap year
debs@deb-socs:~/CIS1300/Labs/Lab2$
```