CS262, Lab assignment 2:
Simple I/O and Credit Card Interest
Due: Sunday, Feb 20 at 11:59 pm ET

Description:

The purpose of this assignment is to code a C program that calculates the amount of interest accrued on a credit card on its expenses.

Preparation:

- Review how to use `fgets()`, `sscanf()`, and `printf()`. Pay attention to how floating point values are printed with `printf()` and how to create a reference to a variable when using `sscanf()`.
- Review how to use the `pow()` function and how to include the `math` library in `gcc`.

Instructions:

The program prompts the user for:
- The total amount of the purchases on the credit card
- An **A**nnual **P**ercentage **R**ate (APR)
- Number of days the money is borrowed

The program will calculate:
- The amount of the compound interest for the purchases

The program prints:
- The original amount of the purchases on the credit card
- The APR
- Number of days the money is borrowed
- *The amount of the interest*
- *Total amount needed to be paid back*

Specifications:

The total amount of the purchases and the ARP should be read as type `double`. The user may or may not enter cents as part of the input, so you might expect values like `500.1`, `330`, `415.7`. The number of days the money is borrowed must be read as type `int` (or `unsigned int`). Ensure that the floating-point output always displays exactly two decimal places.

For interest calculation, we will use the compound interest method, in which, we consider not only the interest on the actual borrowed amount, but also the interest of interest that has already accrued.

To compute compound interest, use the equation:

$$P' = P\left(1 + \frac{r}{n}\right)^{nt}$$

Where:

P	is the original sum
P'	is the new sum
r	is the annual interest rate
n	is the compounding frequency (we assume it to be `12`, like most US banks)
t	is the overall length of time the interest is applied (expressed in years).

The total compound interest generated is the final value minus the initial principal:

$$I = P' - P$$

In order to compute the power function, include the `math.h` library and call the `pow()` function. Note that, the number of days money is borrowed needs to be converted to number of years. If you simply try `days/365` to express it in terms of years, the result of the division may be wrong, at least you make sure to keep the *float* part. The annual interest rate (`r`), can be obtained from the APR, using `r = APR/100`.

Your program must check to ensure that negative values are not input. If an improper value is entered, an error message is printed and the program will exit (do not re-prompt the user for a correct value).

For input, you **must use** the `fgets()/sscanf()` combination. Do not use the `scanf()` or `fscanf()` functions. Although using `scanf()` for retrieving numerical data is convenient, it is problematic when used in conjunction with character string input (as you may discover if you use it in later labs). Therefore, it is best to get into the habit of avoiding `scanf()` completely, and using the `fgets()/sscanf()` combination exclusively in this course.

Makefile:

Your source code will be named `lab2_<username>_<labsection>.c` and you will use a Makefile similar to the one used on Lab 1 to create the executable, but this time your Makefile will include variables and other changes. In this Lab, the `math.h` library is included in your code; therefore, you need to add the flag "`-lm`" when compiling with `gcc`. This flag tells the linker to include the precompiled object for the `math.h` library when creating the final executable.

In your Makefile, include comments similar to those in Lab 1. After the comments and before the `all` target, include these lines:

```
CC = gcc
CFLAGS = -g -Wall -lm
```

Specify the prerequisite for the `all` target and type this command inside the target:

```
    $(CC) lab2_<username>_<labsection>.c -o lab2_<username>_<labsection> $(CFLAGS)
```

Include a target named `clean` to remove the executable.

* During your Lab session, your GTA will explain more details about `CC` and `CFLAGS`

Submission:

You will submit a tarfile containing a directory with your source file, Makefile and a typescript file that shows a listing of your code and that it compiles and runs without errors.

Create the directory `lab2_<username>_<labsection>` inside your `CS262` directory. The directory for Lab 2 must contain your source file and Makefile. Use the `cd` command to go into the `lab2_<username>_<labsection>` directory to start the typescript.

1. Create a typescript file named *lab2_typescript_<username>_<labsection>*
2. Show that you are logged onto zeus. Type: *uname -a*
3. Show you are in your homework directory (*pwd* command)
4. Show a listing of the current directory (*ls* command)
5. Show a listing of your code (*cat* command)
6. Show a listing of your Makefile
7. Remove any executable version that may appear (*make <target>*)
8. Compile the code using *make*
9. Show that the executable was created (*ls* command)
10. Run the executable using the following sets of inputs samples
    Note: Each value is entered individually without commas

```
Run1:    1000, 22.0, 180
Run2:     650, 19.5, 30
Run3:    -470.23, 18.99, 45
Run4:     778.85, 15.99, 65
Run5:     880, -17.99, 70
```

11. End the script command
12. Verify your typescript file is correct (use the cat command)
13. Remove any executable version of your program
14. Create a tarfile of your directory named `lab2_<username>_<labsection>.tar`
15. Copy this tarfile to your local machine
16. Submit the tarfile to Blackboard

Congratulations! You have completed your Lab assignment ☺

Useful material explaining how to calculate `P′` and `I` is available on BB:

https://mymasonportal.gmu.edu/bbcswebdav/pid-14917128-dt-content-rid-232480990_1/xid-232480990_1