COMP161
Lab 3 & Homework 3
Spring 2018

Lab 3 and Homework 3 give you practice with writing and testing basic functional procedures.

Lab 3

For lab you'll be working with the following problem from the first edition of *How to Design Programs*.

Exercise 3.3.1. The United States uses the English system of (length) measurements. The rest of the world uses the metric system. So, people who travel abroad and companies that trade with foreign partners often need to convert English measurements to metric ones and vice versa.

Here is a table that shows the six major units of length measurements of the English system:

English		metric
1 inch	=	2.54 cm
1 foot	=	12 in.
1 yard	=	3 ft.
1 rod	=	5(1/2) yd.
1 furlong	=	40 rd.
1 mile	=	8 fl.

Develop the functions inches→cm, feet→inches, yards→feet, rods→yards, furlongs→rods, and miles→furlongs. Then develop the functions feet→cm, yards→cm, rods→inches, and miles→feet. Hint: Reuse functions as much as possible. Use variable definitions to specify constants.

This problem lists several functions for you to implement[1]. These functions should be declared and defined in a library named *conversions* and in the name space *dist*. *Your goal for lab should be to complete and submit the inches to centimeters conversion. By the end of lab*, submit your source documents[2] and your Makefile as assignment *lab3* using the *handin* program. Do not submit objects, executables, temporary files created by emacs, or any other non-source file. You must setup a clean rule in your Makefile that gets rid of all of this for you so that a simple *make clean* will clean things up for submission.

[1] You cannot use $->$ in function names in C++, so you'll need to adjust the names accordingly. Try something like *feetToInches* or *feet_to_inches*

[2] cpp and h files

Homework 3

Due by class time on 2/7. Submit as hwk3.

1. Complete the remainder of the conversion library functions.
2. Design and develop a procedure for the following problem[3]. This should be in a library named *accounting* and a namespace named *payable*.

[3] again, courtesy of *HtDP1e*

> **Exercise 4.4.3.** Some credit card companies pay back a small portion of the charges a customer makes over a year. One company returns
>
> .25% for the first $500 of charges,
>
> .50% for the next $1000 (that is, the portion between $500 and $1500),
>
> .75% for the next $1000 (that is, the portion between $1500 and $2500),
>
> and 1.0% for everything above $2500.
>
> Thus, a customer who charges $400 a year receives $1.00, which is 0.25 * 1/100 * 400, and one who charges $1,400 a year receives $5.75, which is 1.25 = 0.25 * 1/100 * 500 for the first $500 and 0.50 * 1/100 * 900 = 4.50 for the next $900.
>
> Determine by hand the pay-back for a customer who charged $2000 and one who charged $2600.
>
> Define the function pay-back, which consumes a charge amount and computes the corresponding pay-back amount.

Practice Problem Set

These two problems are probably not enough practice with working the design recipe in C++. Give some serious consideration to working on these problems. You never know, they should show up on a quiz...

Some of these problems might involve Racket strings or Racket symbols. C++ does not have a *symbol* type. I recommend you use a *char* type instead. The limitation you'll need to work with is that a *char* value is a single letter, so you'll have to make due with short[4] names. Alternatively, you can get ahead of the game and look at the C++ *string* type[5]

[4] terrible

[5] `http://www.cplusplus.com/reference/string/string/`

1. From HtDP1e. Problems are listed as *chapter.section.problem* numbers.[6]

   [6] `http://htdp.org/2003-09-26/Book/`

   (a) Basic Functions 3.3.2 – 3.3.5

   (b) Conditionals 4.4.4

2. HtDP Online Problem Sets[7]

   [7] `http://htdp.org/2003-09-26/Problems/`

   (a) (Conditionals) Section 4