CS 111 - Homework 6

Deadline

11:59 pm on **Friday, October 21, 2016**

How to submit

Each time you would like to submit your work:

- Note that, since you are using `funct_play` on nrs-labs to create your C++ functions for this homework, your `.cpp` and `.h` files SHOULD already be on nrs-labs. (Ideally, they are in a folder/directory named `111hw6`.)
- SO -- if you are not already logged onto nrs-labs, then use PuTTY/`ssh` to do so, and use `cd` to change to the folder/directory where your homework files are -- for example,

  ```
  cd 111hw6
  ```

- Use the `ls` command to make sure your desired `.cpp` and `.h` files are really there:

  ```
  ls
  ```

- Use `~st10/111submit` to submit them, with a homework number of **6**
  - Make sure that `~st10/111submit` shows that it submitted **ALL** of your `.cpp` and `.h` files you were intending to submit!

Purpose

To use the design recipe along with `funct_play` to design, write, implement, and test simple C++ functions.

Important notes

- You are expected to use `funct_play` to develop your C++ functions for this assignment.
  - Start early! Then you'll have time to e-mail me your `.cpp` and `.h` files along with the error message(s) if you run into an error you don't know how to handle.
  - (OR -- you can e-mail me those error message(s) and let me know that you've submitted the related `.cpp` and `.h` files using `~st10/111submit` and an unusual homework number, and I can look over your function's files that way in trying to answer your question.)
- If you attended the Week 8 Lab on Friday, October 14, then you should have `expr_play`, `funct_play`, and `funct_compile` on nrs-labs, ready to be run from any nrs-labs folder/directory -- just type one of their names to start them up.
  - If you missed the Week 8 Lab on Friday, October 14, you missed installing these and several other C++ tools you are required to use for CS 111 for the next few weeks.
  - See the posted Week 8 Lab Exercise - Part 1 handout for instructions on how to set these up.
  - See me A.S.A.P. if you have any problems doing so! The deadline will not be extended because you missed lab.
  - And please remember that regular lab attendance is expected for CS 111.

- You are still expected to follow the Design Recipe for all **functions** that you design/define.
  - Remember, you will receive **significant** credit for the signature, purpose, header, and examples/tests portions of your functions.
  - Typically you'll get at least half-credit for a correct signature, purpose, header, and examples/tests, even if your function body is not correct.
  - (and, you'll **lose at least half-credit** if you omit these or do them poorly, even if your function body is correct).
- That said -- remember that, in C++:
  - use **C++ type names** in the signatures of C++ functions
  - write examples/tests as `bool` expressions, typically using == or < . For example,

    ```
    my_funct(3) == 27
    abs(my_dbl(4.7) - 100.43) < 0.001
    ```

    ...and note that these example tests are **EXPRESSIONS** rather than C++ statements -- do **NOT** end them with a semicolon!
- Be especially careful to include at least one specific example/test for each "kind"/category of data, and (when there *are* boundaries) for boundaries between data. You can lose credit for not doing so.
- Note that the C++ `cmath` library, included by `funct_play` by default, includes such goodies as an absolute value function (`abs`), `sqrt`, `pow`, and more.
- Be sure to indent your `return` statement by at least 3 spaces inside your function body!

  And curly braces go on their own line, lined up as shown in the posted class examples!

Problem 1

- Use PuTTY/`ssh` to connect to nrs-labs.
- Make and protect a Homework 6 directory using the commands:

  ```
  mkdir 111hw6
  ```

  ```
  chmod 700 111hw6
  ```
- Change to that directory using:

  ```
  cd 111hw6
  ```

Recall the function from Homework 2, Problem 1 that returns the perimeter of a rectangle. Use `funct_play` to develop a C++ version of this function named `rectangle_perim`. (`rectangle_perim` expects the length and width of a rectangle, and returns the perimeter of that rectangle.)

Submit your resulting `rectangle_perim.cpp`, `rectangle_perim.h` , and `rectangle_perim_ck_expect.cpp` files.

Problem 2

Recall the function from Homework 2, Problem 2 that asks a given person how they are doing. Use `funct_play` to develop a C++ version of this function named `ask_how_doing`. (`ask_how_doing` expects a person's name, and returns a customized question, including that person's name, asking how they

are doing)

Submit your resulting `ask_how_doing.cpp`, `ask_how_doing.h`, and `ask_how_doing_ck_expect.cpp` files.

Problem 3

Recall the function `total-feet` from Homework 2, Problem 4 that computes the total number of miles in a given number of miles and feet. Use `funct_play` to develop a C++ version of this function named `total_feet`. (`total_feet` expects a number of miles and a number of feet, and returns the total number of feet.)

MAKE SURE that you answer `y` and then declare `FEET_PER_MILE` to be an appropriate named constant when `funct_play` asks if you want any named constants! And notice that this named constant declaration appears in your resulting `total_feet.h` file.

Submit your resulting `total_feet.cpp`, `total_feet.h`, and `total_feet_ck_expect.cpp` files.

Problem 4

Recall the function from Homework 2, Problem 5 that computes the total cost for an order of Tasty-Waking coffee. Use `funct_play` to develop a C++ version of this function named `order_total`. (`order_total` expects the number of pounds of coffee in an order, and returns the total price of that order, including shipping.)

Remember the three **named constants** that were part of this problem -- here's how I declared them in the Homework 2 posted example solution:

```
(define COFFEE-PRICE-PER-LB 8.75)
(define SHIP-PRICE-PER-LB 0.75)
(define SHIP-COST-FIXED 2.5)
```

MAKE SURE that you answer `y` and then declare C++ versions of these named:

`COFFEE_PRICE_PER_LB`

`SHIP_PRICE_PER_LB`

`SHIP_COST_FIXED`

to be appropriate named constants when `funct_play` asks if you want any named constants! And notice that these three named constant declarations appear in your resulting `order_total.h` file.

Submit your resulting `order_total.cpp`, `order_total.h`, and `order_total_ck_expect.cpp` files.

Problem 5

Recall the function from Homework 3, Problem 2 that computes the equivalent Celsius temperature for a Fahrenheit temperature. Use `funct_play` to develop a C++ version of this function named `fahr_to_cels`. (`fahr_to_cels` expects a temperature given in Fahrenheit, and returns the equivalent Celsius temperature.)

Submit your resulting `fahr_to_cels.cpp`, `fahr_to_cels.h`, and `fahr_to_cels_ck_expect.cpp` files.