CS 111 - Homework 7

Deadline

11:59 pm on **Friday, October 28, 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 are already on nrs-labs. (Ideally, they are in a folder/directory named `111hw7`.)
- 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 111hw7
  ```

- 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 **7**
  - 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, most involving C++ conditional statements, and one that just uses another function.

Important notes

- Be sure to follow the course coding style discussed in class and demonstrated in posted examples. In particular, remember to indent conditional statements as shown and demonstrated in posted examples.
- 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 a bit longer.
  - 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.

Problem 1

- Use PuTTY/`ssh` to connect to nrs-labs.
- Make and protect a Homework 7 directory using the commands:

  ```
  mkdir 111hw7
  chmod 700 111hw7
  ```
- Change to that directory using:

  ```
  cd 111hw7
  ```

Recall the function from Homework 4, Problem 1 that gives discounts to frequent shoppers based on their level. As a reminder:

A store gives discounts to frequent shoppers based on their past level of purchases; they are either "silver" level, "gold" level, or "platinum" level. Silver level frequent shoppers receive a 12% discount, gold level frequent shoppers receive a 25% discount, and platinum level frequent shoppers receive a 33% discount. All other shoppers receive no discount.

Use `funct_play` to develop a C++ function `discount_amt` that, given a string representing the level of frequent shopper, returns the appropriate discount for that level written as a decimal fraction. (It should return a discount of 0 if the shopper level is not one of those noted above.) You are expected to make and use a named constant for each of the three shopping-level discount values, and make sure you provide sufficient examples/tests.

Submit your resulting `discount_amt.cpp`, `discount_amt.h`, and

discount_amt_ck_expect.cpp files.

Problem 2

THIS FUNCTION DOES **NOT** REQUIRE an if or switch statement!

Recall the function from Homework 4, Problem 2 that used Homework 4, Problem 1's function to determine the discounted total for a purchase by a frequent shopper.

Use funct_play to develop a C++ function total_wi_discount that expects the total of a purchase before discount and a string representing the level of frequent shopper, and produces the appropriate discounted total for that purchase. **For full credit, your solution must appropriately use your** discount_amt **function from Problem 1** - that is, you'll want to answer **y** to funct_play's question:

```
Are there any already-created C++ functions (in the current
   working directory) which you would like to be able to use
   within your new function?
   (type y if so, n if not)
your answer:
```

...and, when it asks, be sure to say that discount_amt is a function you want to use in total_wi_discount.

(Hint: you should **not** need an if statement in this particular function, thanks to discount_amt.)

Submit your resulting total_wi_discount.cpp, total_wi_discount.h, and total_wi_discount_ck_expect.cpp files.

Problem 3

Recall the function from Homework 4, Problem 3 that recommends what outerwear to wear on a given day given the predicted high temperature in Fahrenheit degrees, based on the following:

```
<= 32 - "parka"
(32, 50) - "coat"
[50, 60) - "jacket"
>= 60 - "no outerwear today"
```

Use funct_play to develop a C++ function outerwear_choice that that expects the predicted high temperature in Fahrenheit, and returns the recommended outerwear for that day. For full credit, make sure that you include at least the minimum required examples/tests for this data (hint: including boundary cases!).

Submit your resulting outerwear_choice.cpp, outerwear_choice.h, and outerwear_choice_ck_expect.cpp files.

Problem 4

Recall the function from Homework 4, Problem 4 regarding pizza consumption and exercise.

Use funct_play to develop a C++ function workout_amt that determines the number of hours of exercise required to counter the excess fat from eating pizza. workout_amt expects a number that represents daily pizza consumption, in slices, and produces a number, in hours, that represents the amount of exercise time that you need.

For a daily intake of :	You need to work out for :

0 slices	1/2 hour
1 to 4 slices	1 hour
>4 slices	1 hour + (1/2 hour per slice above 4)

Be sure to include an appropriately complete set of specific examples/tests.

Submit your resulting `workout_amt.cpp`, `workout_amt.h`, and `workout_amt_ck_expect.cpp` files.

Problem 5

Assume that coins are represented as follows:

- 'Q' or 'q' -- quarter
- 'D' or 'd' -- dime
- 'N' or 'n' -- nickel
- 'C' or 'c' or 'P' or 'p' -- cent/penny

Use `funct_play` to develop a C++ function `get_worth` that expects a character representing a coin, **and uses a C++ `switch` statement** to return the decimal worth of that coin (for example, a cent/penny is worth `0.01`). If it receives any character besides those noted above, it should return a worth of `0.0`.

Submit your resulting `get_worth.cpp`, `get_worth.h`, and `get_worth_ck_expect.cpp` files.

Problem 6

This problem's purpose is to provide more practice with the C++ `switch` statement.

Fun fact: the C++ `cmath` library has a function `pow` that expects two `double` arguments and returns the result of raising the first argument to the power given by the second argument. That is, `pow(2.0, 3.0)` gives you the result of raising 2.0 to the power 3.0, and so results in 2.0 * 2.0 * 2.0 == 8.0.

Consider: a character `'+'` cannot be actually used to add two numbers together in C++ -- but if you were given that character, and two numbers, you could write logic that would see if the given character was a '+', and if that is so, then it would add those numbers together using a proper + operator.

Use the design recipe to develop a C++ function `do_op` that indeed expects an operator expressed as a character and two numbers, **and uses a C++ `switch` statement** to return the result of performing the operator corresponding to that character to those two numbers. It should be able to support:

`'+'` -- add the two numbers

`'-'` -- subtract the two numbers

`'*'` -- multiply the two numbers

`'/'` -- divide the two numbers

`'^'` -- raise the first number to the power of the second number (No, C++ does NOT have a ^ operator. But it does have that `pow` function mentioned earlier... 8 -))

(You may add additional operator-characters if you would like.)

It should simply return 0.0 if called with any unsupported/unexpected character as the operator.

For example:

`do_op('+', 3.4, 1.6) == 5.0`

`do_op('-', 5, 2) == 3.0`

(although, if necessary,

`abs(do_op('+', 3.4, 1.6) - 5.0) < 0.01`

`abs(do_op('-', 5, 2) - 3.0) < 0.01`

... `8-)`)

Submit your resulting `do_op.cpp`, `do_op.h`, and `do_op_ck_expect.cpp` files.