CS 112 - Homework 5

Deadline

11:59 pm on **Friday, October 7**

Purpose

To answer questions related to pass-by-reference parameters and pointer basics, to practice writing another function with pass-by-reference parameter(s), to practice a bit with pointers and dynamic allocation, and to practice declaring and using an array of objects.

How to submit

You will complete **Problems 1 and 2** on the course Canvas site (short-answer questions on C++ pass-by-reference parameters and pointer basics).

For **Problems 3 onward**, you will create the specified `.cpp` and `.h` files on the CS50 IDE, and then submit those to the course Canvas site.

NOTE: While I list the separate files you need to submit for each problem below, I am going to set up Canvas to *also* accept `.zip` files.

That is,

- you can submit each `.cpp`, `.h`, and `.txt` file to Canvas
- OR, if you prefer, you may compress your files to be submitted into a single `.zip` file and submit that `.zip` file to Canvas.

Problem 1 - 6 points

Problem 1 is correctly answering the "HW 5 - Problem 1 - Short-answer questions on pass-by-reference parameters" on the course Canvas site.

Problem 2 - 10 points

Problem 2 is correctly answering the "HW 5 - Problem 2 - Short-answer questions on pointer basics" on the course Canvas site.

Problem 3 - more pass-by-reference practice

Consider the following three functions (which I hope look familiar!). Choose one of these, and implement it in its own `.h` and `.cpp` files. Also implement an appropriate testing `main` for your choice in its own `.cpp` file.

(It is fine if, for additional practice, you implement more than one of these.)

Problem 3 - option 1

A function `get_user_answer` expects a pass-by-reference `char` parameter, and has the side-effects of asking the user to enter either a, b, c, or d, and then reading in what they enter -- BUT if they do not enter one of those four character values, it KEEPS asking until they DO. Then it CHANGES the pass-by-reference parameter to their "good" entry, and returns nothing.

Problem 3 - option 2

A function `put_in_order` expects two pass-by-reference `double` parameters, has the side-effects of seeing

if the first parameter is greater than the second parameter, and if so, it swaps their values, and returns nothing.

Problem 3 - option 3

A function `min_and_max` expects four arguments:

- a pass-by-value argument, an array of `double` values (that will NOT be changed by this function),
- a pass-by-value argument, that array's size,
- a pass-by-reference argument of type `double`, which will be set by this function to be the smallest value in the given array, and
- and another pass-by-reference argument of type `double`, which will be set by this function to be the largest value in the given array,

...and returns nothing.

Submit your resulting `.cpp` and `.h` files.

Problem 4

We'll see that pointers get used a lot with dynamic arrays and with a data structure, linked lists, that we will be discussing. By themselves -- not as much!

But, you should have a chance to practice with pointers and dynamic allocation a bit on this homework, so:

Write a `main` function in a file `hw5-prob4.cpp` that does at least the following:

- declares at least three pointer variables, one whose type is pointer-to-an-`int`, one whose type is pointer-to-a-`double`, and one whose type is pointer-to-a-`string`.
- Write statements to cause each of them to point to dynamically-allocated memory.
- Write statements to set the memory each is pointing to a non-zero/non-empty value of your choice.
- Write statements to print to the screen, in a tasteful manner, the values that each of your pointers is pointing to.
- Write statements to free/deallocate the dynamic memory each is pointing to.

Submit your resulting `hw5-prob4.cpp` file.

Problem 5

Consider your C++ class **`PlayingCard`** or **`GameCard`** from Homework 4.

(Note: since you also will be submitting its `.h` and `.cpp` files for this homework, it is fine if you have improved your **`PlayingCard`** or **`GameCard`** class since the version you submitted for Homework 4, as long as it still meets Homework 4's minimum requirements. Just make sure that the version you submit with this homework works with the class you create here.)

ALSO consider the posted handout "FUN FACTS about USING a user-defined class", posted along with this homework handout.

Write a `main` function in a file `hw5-cards.cpp` that does at least the following:

- Declares an array of at least 4 elements of your card class from Homework 4.
- Sets each of the elements in this array to contain cards with data fields at least somewhat different from each other. (That is, the card objects should NOT all be equivalent to each other!)

  - It is up to you how the cards' values are determined:
    * You can "hard-code" in their values (as I did for `team[0]`, `team[1]`, and `team[2]` in the "FUN FACTS about USING a user-defined class" handout).
    * You can ask the user to enter values for the cards' values, and use what they enter in appropriate constructor calls.
    * You can read the values for the cards' values from a file, and use what is read in appropriate constructor calls.
- Uses an appropriate loop to show the resulting contents of the array.
  - You get to decide if you'd like to call its `display` method for each card object in the array,

    or to `cout` the results of its `to_string` method,

    or to `cout` the results of a tasteful selection of its accessors methods, etc.!
- If you'd like to do anything else with your array now that you have it, that's fine!

Submit your resulting `hw5-cards.cpp` along with the `.cpp` and `.h` files for your card class, and, if applicable, include any other `.cpp`, `.h`, and `.txt` files used by your program.