CS 112 - Homework 6

Deadline

11:59 pm on **Friday, October 14**

Purpose

To answer questions related to pointers to an object and dynamic arrays, to practice more with dynamic arrays of objects.

How to submit

You will complete **Problems 1 and 2** on the course Canvas site (short-answer questions on pointers to an object and `bool` expressions).

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 - 12 points

Problem 1 is correctly answering the "HW 6 - Problem 1 - Short-answer questions writing statements involving pointers to an object" on the course Canvas site.

Problem 2 - 10 points

Problem 2 is correctly answering the "HW 6 - Problem 2 - Short-answer questions on dynamic arrays" on the course Canvas site.

Problem 3 - using a dynamic array of `Point` objects

Consider your `Point` class from the Week 6 Lab Exercise.

Write a `main` function in a file `points-play.cpp` that uses this `Point` class to do the following:

- Asks the user how many points they wish to enter, and creates a dynamic array of that many points.
- Read in from the user the x and y coordinates for that many points, assigning corresponding `Point` objects to the dynamic array.
- Then: choose one of the following options below.
- Whichever option below that you you choose, don't forget to then free that dynamically-allocated array's memory when you are done using it! (And for good measure and for practice, set the pointer involved to `NULL` after doing so.)

Problem 3 - Option 1

Think of the entered points in that array as locations on a path. Then, using the `Point` class' `dist_from`

method, compute the total length of that path, from the first point in the array to the last point in the array, and print that path's total length to the screen in a tasteful descriptive message.

Problem 3 - Option 2

Think of the entered points in that array as locations of interest -- they might be locations of magic items, or location of electric car chargers, or locations of team members or allies, or locations of bathrooms or water fountains, etc.

Them, ask the user to enter the x and y coordinate of their current position, and then, using the `Point` class' `dist_from` method, determine which point in the array is closest to the user's current position, and print, in a tasteful descriptive message, the x and y coordinate of that closest point, and also the distance from the user's location to that closest point.

Submit your file `points-play.cpp`, `Point.h`, and `Point.cpp`.

Problem 4 - using a dynamic array of card objects

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

Write a `main` function, in a file `cards-play.cpp`, that meets at least the following minimum requirements:

- It should ask the user how many cards they want, and dynamically allocate an array able to hold that many instances of your **`PlayingCard`** or **`GameCard`** class.
  - ALTERNATIVE option: it can ask the user for the name of an input file, assumed to start with one line containing the number of cards' information it has followed by the card's information, one card per line. It should then read the number of cards from the file and dynamically allocate an array able to hold that many instances of your **`PlayingCard`** or **`GameCard`** class.
- Interactively ask the user to enter that many cards' worth of information into your array.
  - ALTERNATIVE option: read the cards' information from the file into your array.
- Call either your method `display` or `to_string` for each card object in the resulting array, to show its current contents.
- Do SOMETHING of your choice to change those cards -- for example:
  - you could increase the power of each card by some amount or percentage
  - you could change the suits of each card in some way (changing heart to diamond, diamond to heart, spade to club, and club to spade, for example)
  - you could increase the value of each card by one modulo the maximum possible (such that a "top" valued card "rolls over" to the lowest value, then -- a K or 12 might become an A or 1, for example)
  - you could have a loop that asks the user if they want to change any cards, and if so, it allows them to specify which card to change, and how they want it changed
  - you could have a loop that asks the user for information about a card, and that card is somehow compared to each element in the array, possibly replacing one of them (it might replace the first card that it is somehow "better" than, based on a criteria you choose -- the first it has a higher value than, for example)
  - (or you can make some other change(s) you choose)
- When done with those change(s), again call either your method `display` or `to_string` for each card object in the resulting array, to show its resulting contents.

- Don't forget to then free that dynamically-allocated array's memory! (And for good measure and for practice, set the pointer involved to `NULL` after doing so.)

Submit your resulting `cards-play.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.