Pair Programming 10.1 Activities

* **Always use the pair programming tests to ensure your program works properly.**
* **Take a screen shot of each execution in the tests.**
* **Download the source code file for inclusion in the turn in document.**
* **Turn in pair programming activities using the pair programming turn in document.**
* **It is each individual’s responsibility to turn in the assignment and pair programming is graded individually so make sure you share the work you and your partner did together as you go.**
* **Do not share work with your partner(s) if they did not work together with you on the lab.**

10a: (2 points) Write a C++ program in a file named pp10a.cpp that creates a structure typed named Pet that has a name, an age, a weight (in lbs) and whether the pet has been neutered or not, *isNeutered*. Write a main function that declares a Pet variable, asks the user for a name, an age, the pet’s weight and whether the pet has been neutered. Store this data in the Pet variable data members and display the data from the Pet variable. In order to set the Boolean data member *isNeutered*, ask the user to answer Y for yes or N for no, then set this Boolean based upon the character the user entered (be sure your program accepts either an upper or lowercase ‘y’ or ‘n’. Similarly, when displaying the pet’s data, use the Boolean variable to decide what to print.

10b:(2 points) In a C++ file named pp10b.cpp, write the following two functions. Write a main function that calls these functions to get one pet’s information and print it.

* getPet: gets a pet’s data from the user. This function has one Pet struct parameter. The post-condition is that this parameter has been altered to include the pet’s data.
* displayPet: displays a pet’s data. This function has one parameter, a Pet struct. The formal parameter should be a reference variable so that the actual parameter is not copied to the formal parameter to be efficient. It should be a constant to be safe since the displayPet function should not be allowed to alter the actual parameter (which it could if the parameter was a non-constant reference variable).

The output result of 10b will be essentially the same as 10a, but we’ve made our code more modular in 10b to isolate the “job” of each function.

10c**:** (3 points) In a file named pp10c.cpp, write a function named displayPets that has two parameters, an array of Pet structures and the number of valid elements in the array. In this function, use a for loop to display all of the names, ages, weights and neuter status for all pets in a nicely formatted table as shown in the tests. Write a main driver that has an array of 5 Pet structures initialized as shown below that calls the displayPets function to print the pet data.   
Pet pets[5] = { {“Spot”, 6, 40.2, true}, {“Lassie”, 1, 14.5, false}, {“Loner”, 1, 8.2, false}, {“Brutus”, 11, 92.4, true}, {“Yetti”, 16, 28.7, true} };

10d: (3 points) Copy pp10c.cpp to a file named pp10d.cpp. Add a sortPets function to this file that has two parameters, the array of Pet structures and the number of valid elements in the array. Write code to implement a sorting algorithm such as selection sort to sort the pets by name. The post-condition is that the array of pets has been reordered. Update the main driver to call sortPets before displaying the pet data. **Display of the sorted data should occur in the main function, NOT inside the sortPets function.**