CSIS 6033 – Lab 2

Defensive Programming

In this lab, you will be completing a programming exercise through the point of view of both a contracted library developer and the client that will use the developed code.

In the first part, you will be required to write a class in C++ that will be included in the client's code. Your class must be written with defensive programming in mind. It should allow the client to include or leave out your defensive checks at compile-time. In the second part, you will use your defensively programmed class methods to guide the revision of the provided user driver program. After encountering failures from misuse of the library class methods, you will update the driver program, according to the implementation guidelines in the CWE documentation for the appropriate error.

Note that the code you write does not have to be completely optimized and you will likely see better ways to write the class and the driver to avoid the problems inherit in the client descriptions.

In Part 1 of the lab you will complete the following:

- Write a class called myArray in a file named "yourlastname_lab2.cpp" where you substitute your own name
    - Constructor
        - two inputs: int size and string input
        - dynamically create a char* array of int size
        - parse string input (which should be a string of comma-separated characters) and enter the characters in the array in order
    - Destructor should free the memory assigned to your char* array
    - ReadFromArray
        - one input: int index
        - return char at the given index for the array
    - WriteToArray
        - two inputs: int index, char replace
        - overwrite char at given index with new char replace
    - DeleteArray
        - free the memory for your char*
        - set char* to NULL
    - PrintArray
        - output the contents of the char* array to stdout
    - NewArray
        - two inputs: int size and string input
        - dynamically create a char* array of int size
        - parse string input (which should be a string of comma-separated characters) and enter the characters in the array in order
- For each class method, provide the contract for proper usage of the method
    - enter as comment lines directly after the definition
    - List any preconditions (what has to be true immediately before executing the method)

    - List any postconditions (what has to be true immediately after executing the method)
- Utilize C standard assert() library calls from assert.h to test your preconditions
- Use macros to give the client the option on whether to include the asserts at compile-time
- Use the provided sample client driver program to test your class code
- **Take screenshots of your assertions being invoked for each function**

In Part 2 of the lab you will complete the following:

- Using the assertions you have placed into your class methods, update the driver code to ensure calls made to the class methods are in-contract
- Identify what CWE errors, if applicable, are occurring with out-of-contract use of your class methods
- Review the 'Potential Mitigation" section for those CWE errors and use the "Phase: Implementation" entries to guide your revision of the provided program driver.
- **Take screenshots of the driver code working without hitting the assertions** – be sure to explain in your word document how you tested the preconditions of each method and what changes you made to the driver to ensure in-contract calls were made to the methods.

Graduate students should also answer the following:

- Is there a Python equivalent to the C-standard assert() calls used in class with C++?
- How would you approach defensive programming from the point-of-view of python methods?

Submit a zip file to Blackboard which contains your class file and a word document which includes the screenshots and other information described above.

For full credit your code should compile, run as described, and be appropriately commented. If I need to know anything in particular about how I should compile your code, include that in your document.

You should submit your solutions to Blackboard by **Thursday, July 9 at 11:59pm**. Your assignment should be of the format CSIS6033_Lab2_pape.zip, where you should substitute your own last name. Be sure to clearly indicate any referenced material that you used to complete this assignment.