Homework #7 – Fun with Strings and Vectors

In this assignment you are asked to implement a `StringUtility` class containing a variety of member functions that work with `vectors` of `strings`. The following UML class diagram shows the attributes and behaviors of class `StringUtility`. ***Important:*** *In this assignment you may not use arrays or the C string handling functions (e.g. `strlen`, `stcpy`, etc.); you may only use `vector` and `string`.*

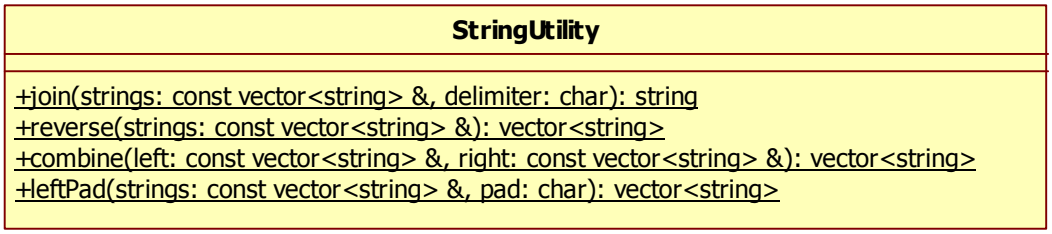

Figure 1. UML class diagram for class StringUtility

1. **(1 point)** Meet these basic requirements:
   a. All non-test code must be implemented in a namespace based on your first and last name (e.g. "`RayMitchell`").
   b. The `StringUtility` class must be defined in a file named "`StringUtility.h`"; all member functions must be defined in a file named "`StringUtility.cpp`".
   c. The `main` function and tests demonstrating must be placed in a file named "`hw7.cpp`".
   d. Make sure `const` is used correctly everywhere within class `StringUtility`. Be sure to check all pointer parameters, reference parameters, and member functions for proper "const-ness".

2. **(1 point)** Define member function `join` as shown in the UML diagram. `join` should concatenate the strings from the `strings` parameter together placing the `delimiter` character in between each pair of strings. The resulting string should be returned. For example, if `join` is called with `{"abc", "def", "ghi"}` and `'*'` the returned string should contain `"abc*def*ghi"`.

3. **(1 point)** Define member function `reverse` as shown in the UML diagram. `reverse` should return a vector containing the strings from the `strings` parameter in reversed order and with the contents of each string reversed. For example, if `reverse` is called with `{"abc", "def", "ghi"}` the returned vector should contain `{"ihg", "fed", "cba"}`.

4. **(1 point)** Define member function `combine` as shown in the UML diagram. `combine` should return a vector containing every string from `left` concatenated with every string from `right`. For example, if `combine` is called with `{"ab, "cd", "ef"}` and `{"gh", "ij", "kl"}` the returned vector should contain `{"abgh", "abij", "abkl", "cdgh", "cdij", "cdkl", "efgh", "efij", "efkl"}`. *Note, the order of the combined strings in the returned vector isn't important as long as all combinations are present.*

5. **(1 point)** Define member function `leftPad` as shown in the UML diagram. `leftPad` should return a vector containing each string from the `strings` parameter left-padded with the `pad` character so that each resulting string has a length equal to the longest original string. For example, if `leftPad` is called with `{“a”, “bb”, “ccc”}` and `‘*’` the returned vector should contain `{“**a”, “*bb”, “ccc”}`.

6. **(1 point)** Implement a test demonstrating `join` working properly with the following arguments:

```
strings → {“The”, “quick”, “brown”, “fox”, “jumps”, “over”, “the”, “lazy”, “dog”}
delimiter → ‘,’
```

Your test must output the contents of the string returned from `join`.

7. **(1 point)** Implement a test demonstrating `reverse` working properly with the following arguments:

```
strings → {“The”, “quick”, “brown”, “fox”, “jumps”, “over”, “the”, “lazy”, “dog”}
```

Your test must output the contents of the vector returned from `reverse`.

8. **(1 point)** Implement a test demonstrating `combine` working properly with the following arguments:

```
left → {“Mr.”, “Mrs.”}
right → {“Jones”, “Smith”, “Williams”}
```

Your test must output the contents of the vector returned from `combine`.

9. **(1 point)** Implement a test demonstrating `leftPad` working properly with the following arguments:

```
strings → {“The”, “quick”, “brown”, “fox”, “jumps”, “over”, “the”, “lazy”, “dog”}
pad → ‘*’
```

Your test must output the contents of the vector returned from `leftPad`.

10. **(1 point)** Make sure your source code is well-commented, consistently formatted, uses no magic numbers/values, follows a consistent style, and is ANSI-compliant.

Place all source code and a screen capture of the output produced by your program in a single Word or PDF document. Submit this document.