EEL3834 - Programming for Electrical Engineers
Fall 2016
Programming Assignment 9
Assigned: 11/26/2016 Due: 12/07/2016 @ 11:59PM
To be done individually

Programming Project 6 from Chapter 10 of Absolute C++ 5th ed. (Savitch), pg. 469:

6. One problem with dynamic arrays is that once the array is created using the new operator the size cannot be changed. For example, you might want to add or delete entries from the array similar to the behavior of a `vector`. This project asks you to create a class called `DynamicStringArray` that includes member functions that allow it to emulate the behavior of a `vector` of strings.
The class should have the following:

1. A private member variable called **`dynamicArray`** that references a dynamic array of type string.
2. A private member variable called **`size`** that holds the number of entries in the array.
3. A default constructor that sets the dynamic array to `NULL` and sets `size` to 0.
4. A function named **`getSize`** that returns `size`.
5. A function named **`addEntry`** that takes a string as input. The function should create a new dynamic array one element larger than `dynamicArray`, copy all elements from `dynamicArray` into the new array, add the new string onto the end of the new array, increment size, delete the old `dynamicArray`, and then set `dynamicArray` to the new array.
6. A function named **`deleteEntry`** that takes a string as input. The function should search `dynamicArray` for the string. If not found, it returns false. If found, it creates a new dynamic array one element smaller than `dynamicArray`. It should copy all elements except the input string into the new array, delete `dynamicArray`, decrement size, and return `true` .
7. A function named **`getEntry`** that takes an integer as input and returns the string at that index in `dynamicArray`. It should return `NULL` if the index is out of `dynamicArray`'s bounds.
8. A copy constructor that makes a copy of the input object's dynamic array.
9. Overload the assignment operator so that the dynamic array is properly copied to the target object.
10. A destructor that frees up the memory allocated to the dynamic array.

This class should be used in a program you write. To give you practice with separate compilation, you will need to split up your program into two files. Call them '**DynamicStringArray.h**' and '**DynamicStringArray.cpp**'. Take note of the capitalization and spelling of the filenames and functions mentioned above. This is **VERY IMPORTANT**, since for this assignment I am not asking you to submit a test file. Your class will be run with a test file that exercises all the functions in the '**DynamicStringArray.h**' file; therefore, class name and functions in the '**DynamicStringArray.h**' file must match up perfectly for it to compile correctly.

Your grade will be subject to the following condition(s):

- Submission:
  The submission deadline is **11:59PM** on **12/07/16**. You will be penalized in increments of 25% per day late (regardless of the time).

  Submit your code on Canvas. You need to upload a zip file containing two files: 'DynamicStringArray.h' and 'DynamicStringArray.cpp'. Also, PLEASE double check your submission to make sure the file has been correctly uploaded.

Your grade will be calculated based on the following (total 10 points)

- Compilation: **2 pts**
  Your code MUST compile in a Linux environment. Since that is the environment in which it will be graded. There is no partial credit available here, either your code compiles or it doesn't.

- Execution/Correctness: **6 pts**
  Your program will be tested with something similar to the test output. In addition, it should have the 10 listed elements in the assignment description. This means that if your program seems to work but you have not actually implemented/included things in the list, you will lose points.

- Style/Organization: **2 pts**
  Your code will also be graded on its style. This includes things like using meaningful variable names, useful comments, proper indentation and spacing, and the proper use of functions. Proper use of functions means wrapping up code that is used in multiple parts of your code in a function. All of these things make your code easy to read and maintain. Partial credit will be available here. As a minimum, your code should have a comment at the beginning with your name, date, and a high level but still descriptive overview of what the program does.

  Pay attention to issues of programming style:

  - **use indentation**
  - **comment your code/methods**
  - **use meaningful names for variables**
  - **leave spaces between logical blocks of the code**
  - **use functions properly**