Marble Pre-Processor

Due Date:	2019-JULY-08	Late Work:	`Not Accepted`
Extra Credit:	2019-JULY-01	Amount:	`+25%`

1 Overview

In his *Future of Programming* lecture, Robert Martin briefly touches on the early development of both C++ and Objective-C. In each case, developers used the tools they had to build the ones they wanted. This technique persists in modern programming, as several intelligence leaks confirm, for the Central Intelligence Agency built frameworks [1] around the C++ language to simplify development and deployment of sensitive tools.

In this assignment, students shall follow in these footsteps and create a specialized pre-processor designed to help conceal strings in executable files using the C++11 programming language. This extensive project requires students create both a unique class to represent strings, but they must also build the application which modifies arbitrary C++ input files by replacing certain string calls with calls to the student-generated class.

2 Requirements

This project requires students submit multiple C++11 source files: The marble string object and the program that swaps it into source files. Students must use the object-oriented model when building the string marble object.

2.1 Marble Class

Each submission must include its unique implementation of a marble character object in its own file. This class shall include the following functionality:

- A constructor accepting a string-literal as its only required parameter.
- Override the << operator such that the marble object returns the unscrambled version of the string.
- Implement a `size()` method which returns the true size of the concealed string.

[1] Allegedly.

Original	Modified
`char data[] = "abc";`	`MarbleString data = "bcd";`
`char stuff[ ]="rEd" ;`	`MarbleString stuff = "sFe";`

Figure 1: Example of a modified declaration statement. The trivial marbling algorithm simply adds one to each character.

- Store the concealed string on the stack in a `char` array.

The particular algorithm students use to conceal the original string-literal remains up to the implementer. Trivially, as shown in the examples in this document, one might add one to the original value. This facilitates simple error discovery during development. Alternatively, students may attempt a more complicated approach. Consider bit shifting and adding to the original characters, for one may rapidly reverse these modifications. In any event, the marble class shall not store the original string literal in an unmodified form, and the transformation must be reversible.

When printed, instead of displaying the marbled text, the marble class object shall override the << operator such that it produces the original string. Consequently, the class must use a reversible algorithm when concealing the string. For example, one could hard-code in an 8-bit key to use, and then exclusive-OR each byte of the string literal with this value to produce the concealed string. When it comes time to print this string, one simply performs the exclusive-OR operation a second time to recover the original text.

At run-time, the program can recreate the string, but it shall not appear in the original text.

2.2 Driver Application

The driver application scans an input source file for character array declaration statements matching a particular from, and it then replaces them with calls to a custom class. Additionally, for files it modifies, the application must also include a pre-processor directive inserting the custom class in the modified input file.

The modified call replaces the original string literal with a scrambled version in the modified output source file. Consequently, after compiling the modified source file, the original string literal will not leap out in plain text.

2.2.1 Replaced Strings

Although one may insert string literals in source code several ways, the driver application implemented in this assignment shall only replace character array, string-literal in an assignment, declaration statement. Simple character array declarations, without an accompanying string literal assignment, shall remain untouched. Figure 1 provides examples of the replacement.

Spaces may appear throughout the declaration, and the driver application must accommodate these accordingly, but it need not preserve them in the final substitution. That is, a declaration without a space in the original may insert one in the modified version.

2.2.2 Output File

After replacing the standard calls in the original source file with calls to the custom class, the program shall store the modified output in a new file. The new output file shall use the same path and basic name as the original file, but it shall append a `mbl-` prefix to the name. For example, if the user identified `cryptic.cpp` as the input file, then `mbl-cryptic.cpp` will hold the modified output.

2.3 Coding Style

Software quality strengthens program security, reliability, and maintainability. In general, all submitted software must:

- Adhere to a consistent naming convention and follow best practices for variable naming.
- Use private, or protected, internal methods to help clean the code.
- Remain free of commented out blocks of code.
- Use consistent formatting.
- The maximum column width is 80 characters.

The graders may elect to subtract points as they deem necessary for coding style and naming convention violations.

2.3.1 Input File Format

This driver application scans C++ source files for certain character array declaration statements and substitutes in a custom object.

It operates on a single, user-supplied source file with each program activation, so solutions need not scan recursively for all files in the project. The driver assumes the input file complies with the syntax and formatting defined

3 Delivery

At the start of class on the due date, students shall submit a legible, physical ASCII printout of their source code. Do **not** submit screen-shots of source code or the development environment. Do **not** copy the source into a word-processor. Simply print the source file, which is ASCII, directly. The 80 character line limit in defined in the coding standard ensures that the source file prints as it appears on screen.

For *extra credit*, students shall submit a fully-compliant solution on the earlier due date identified in the prompt header. Students seeking this approach shall **not** receive additional, priority, or expiated assistance from the teaching assistants or instructors.

4 Grading

In general, all submissions must meet every requirement in this specification.

Requirement	Points
Coding Style	10
Application	20
Marble Class	70

5 Other Notes

The instructor and teaching assistants remain available to assist students with questions that arise during the application's development and design. Due to high demand, however, students may not receive attention.

For an example of some advanced algorithms to use, feel free to explore https://wikileaks.org/vault7/document/Marble/.