CPSC 350: Data Structures and Algorithms
Spring 2024

Programming Assignment 1: Robber Language Translation

Due: February 24, 2024, at 11:59 pm

The Assignment

Most of you are probably familiar with the popular language game, Pig Latin. In fact, some of you may also recall a similar game called Double Dutch, which is akin to Pig Latin but with a different set of rules that focus on consonants instead of vowels. It turns out that these types of language games are popular throughout the world. In Sweden, for example, a popular game is Rövarspråket, which is called "The Robber Language" in English.

In this assignment, you will build a program that translates American English text stored in a plain text file to Robber Language/ Rövarspråket. The goal is to exercise your basic C++ skills before moving on to more advanced data structure implementations.

Robber Language Rules (courtesy of Wikipedia)

The formula for encoding is simple. Every consonant (spelling matters, not pronunciation) is doubled, and an o is inserted in between. Vowels are left intact.

For example, the English word "stubborn" translates to Robber Language as ***sostotubobboborornon***

Program Design

Your program should implement the following classes EXACTLY per the specification (including naming and capitalization) below. FAILURE TO FOLLOW THE SPECIFICATION will lead to an automatic 25% deduction in your score. Each class should consist of a .h file and a .cpp file.

The Model Class

You will build a class named Model that will encode the rules of the Robber language. The class will contain the following public methods:

- A default constructor
- A default destructor
- translateSingleConsonant– takes a single consonant character as input and returns a string representing its encoding in Rövarspråket. Capitalization should be preserved.
- translateSingleVowel – takes a single vowel character as input and returns a string representing its encoding in Rövarspråket. Capitalization should be preserved.

The Translator Class

You will build a class named Translator that will translate English sentences to Rövarspråket sentences using the Model class. The class will contain the following public methods:

- A default constructor
- A default destructor
- translateEnglishWord – takes a single string representing a single English word as input and returns a string representing the Rövarspråket translation.
- translateEnglishSentence – takes a single string representing a single English sentence as input and returns a string representing the Rövarspråket translation. Make sure to account for punctuation.

The FileProcessor Class

You will build a class named FileProcessor that will take text files (.txt) containing English text and produce a HTML file containing the equivalent Rövarspråket translation that can be viewed in a standard web browser. The class will contain the following public methods:

- A default constructor
- A default destructor
- processFile – takes a string representing the input file (English) and a string representing the output file (where the Rövarspråket translation will be written). This method has a void return type. The method should produce an HTML file that has the original English text in **bold** followed by an empty line, followed by the Rövarspråket translation in *italics*. If you are not familiar with HTML, a simple tutorial is available here: https://html.com/#Creating_Your_First_HTML_Webpage

Your classes may contain any private methods you wish to implement to achieve the requirements of the public methods. For example, in the FileProcessor, you may want helper methods to produce and format the required HTML tags. String should use the std::string type. (Not c-style char*)

In addition to the above, you should provide a main method in a file called main.cpp. It should only contain a main method. Your main method should do the following:

- Instantiate a FileProcessor
- Translate the provided input file to Rövarspråket using the file processor.
- Exit

Rules of Engagement

- You may **NOT** use any non-primitive data structures. (No arrays, Vectors, Lists, etc) Just use individual primitive variables (int, double, etc) and std strings. Hopefully, this will convince you that data structures make programs more efficient and easier to write. (Though we suspect you know this already…) Of course, to do the file processing you may use any of the standard C++ IO classes.
- For this assignment, you must work individually.

- Develop using VSCode and make sure your code runs correctly with g++ using the course docker container.
- Feel free to use whatever textbooks or Internet sites you want to refresh your memory with C++ IO operations, just cite them in a README file turned in with your code. All code you write, of course, must be your own. In your README please be sure to include the g++ command for compiling your code.

Due Date

This assignment is due at 11:59 pm on 2-24-2024. Submit all your commented code as a zip file to Canvas. The name of the zip file should be **LastName_FirstInitial_A1.zip.**

** Optional: commit/upload your assignment to GitHub. This is an excellent opportunity to build your project portfolio.

Grading

Grades will be based on correctness, adherence to the guidelines, and code quality (including the presence of meaningful comments). An elegant, OO solution will receive much more credit than procedural spaghetti code. I assume you are familiar with the standard style guide for C++, which you should follow. (See the course page on Canvas for a C++ style guide and Coding Documentation Requirements.)

Again, code that does not follow the specification EXACTLY will receive an automatic 25% deduction. Code that does not compile will receive an automatic 50% deduction.