Programming Exercise

The following programming exercise is designed to provide us with a comprehensive look into your practical programming skills. Please read the directions carefully, and feel free to contact us if you have any questions or need any clarifications.

Overview

You should write a program that will read the contents of a file containing a single line of text, perform several tasks on the contents, and output the results of those tasks. For the purposes of this exercise, the input files will follow these rules:

1. All characters within the input file will be ASCII between characters 32 (space) and 126 (~). There will be no non-printable or special characters.
2. A `word` is defined as a single string of sequential alphabetic characters. In particular, a word does not contain any apostrophes (e.g. `let's` should be counted as 2 words: `let`, and `s`), even though this does not match the typical English definition of a word.
3. A `sentence` is defined as a string of characters terminated by a period or the end of the input. If an abbreviation includes several periods (for example, "U.S."), these are considered separate sentences. So, the text "The U.S. flag" has 3 sentences: "The U", "S", and " flag".
4. Capitalization is not considered. For example, "The", and "the" are considered to be the same word, and "Non" is considered to be a palindrome.

Rules

You may:

- Use any mainstream programming language that you prefer to complete the exercise. A free compiler or interpreter must be available.
- Use Google, StackOverflow, or any other online resources to help you solve the problem.
- Use any open-source libraries, as long as you document which you used and for what purpose. You do not need to document the standard library for the language you are using.

You may not:

- Directly copy any code (including individual functions, classes, or snippets) from any other source, including modifying it.
- Use any closed-source libraries, or any open-source libraries that are not documented.
- Use any languages or programming environments that require a license to build or run the code.

Input/Output

You are free to choose a preferred method of loading the file. For example, you might want the filename passed in as a command-line argument, or you might want to open a dialog to navigate to select the correct file. If you are using a web language, you might want to have the file uploaded through a browser. Any of the above is fine, as long as you explain how to pass the file into the program. Note that you may not require that the *c ontents* of the file are passed into the program, by copying and pasting. Additionally, the file path may not be hard-coded, it must be passed into the program at runtime.

Output is also similarly flexible, In the case of a command line program, you might use standard out. If you are writing a GUI program, you might display a window containing the output. In the case of a web application, it might return the output as the webpage.

Tasks

The following tasks are required. The output from each should be clearly separated, although the exact separation character is up to you (for example, you may choose to separate each output with a newline character).

1. Output the number of unique words that appear in the input. For example, with the input text "The quick brown fox jumps over the lazy dog", there are **8** unique words.
2. Output a list of all unique words that appear in the input, sorted alphabetically. For the purposes of this step, you may output the words in lower case, if desired.
3. Determine if any of the unique words are palindromes. If so, print out a list of the words that are. For the purposes of this step, you may output the words in lower case, if desired.
4. Output the average number of letters in each word in the input. For example, with the input text "The quick brown fox jumps over the lazy dog", the average number of letters in each word is **3.89**.
5. For each sentence in the input, output the reverse. You may collapse any whitespace between sentences into a single space. For example, with the input text "The quick brown fox jumps over the lazy dog. The lazy dog jumps over the quick brown fox", the output should be "god yzal eht revo spmuj xof nworb kciuq ehT. xof nworb kciuq eht revo spmuj god yzal ehT"

If you have difficulty with any of the tasks, please include any code you have worked on while trying it, and explain where you got stuck. We will consider partial solutions.

Building/Running

If you choose a language that requires compiling (for example, C or Java), you should provide instructions for how to compile it. If you choose a language that is interpreted, you should provide a description of the environment required to test it (for example, if you need a webserver installed, mention that). If you use any libraries or modules that are not part of a standard installation of that environment, be sure to include a description of where to find them. Additionally, please include a description of how to appropriately run the program.

Evaluation

Where possible, you should deliver code that you consider "production-ready". We will be evaluating your submission based on the following (not necessarily in order):

- Code cleanliness and ease of reading: Code that is clean and easy to read and understand is preferred.
- Runtime efficiency: Efficient code is preferred
- Use of appropriate data structures and algorithms
- Proper error handling
- Comments where appropriate

Submission

You should submit an archive (`.zip`, `.tar.gz`, etc) containing the following:

1. A text file named `README`, containing instructions for building, a list of any libraries you used, and any other information or insight you want us to know.
2. A `src/` directory, containing all source code you wrote for this exercise.
3. (Optional) A `lib/` directory, containing any libraries you used in the project. In general, we prefer that you provide links to the websites of the libraries, but if that is not possible, include them here.

Please do **not** include any compiled code (executables, object files, class files, etc).