Bonus Assignment

ECE 345 Algorithms and Data Structures
University of Toronto
Dept. of Electrical and Computer Engineering
Winter Semester, 2015

Due date: April 3, 2015, 8pm by email to `briank@eecg.toronto.edu`

1 Introduction

This bonus assignment is intended to allow you to learn about some of the practical aspects of algorithms and data structures by solving a programming problem in C++. It is worth a bonus of up to **4%** of your final grade and is *completely optional.* This is an individual assignment (no groups).

2 Problem

You are given an input file with multiple lines where each line contains a list of words separated by a space. All words contain only lowercase letters (`a-z`). Write an efficient C++ program to find and output the *longest consecutive palindromic sequence of words* belonging to *every* line in the file.

Here's an example input file with three lines:

```
here the bottom the here bed is where the where
where the where here the bottom the here bed here know
here where here the bottom the here bed there here
```

Your algorithm should return:

```
here the bottom the here
```

Notice that `here the bottom the here` is the longest consecutive sequence of words (in this case 5 words long) that is part of every line and that is also palindromic. Palindromic in this context is defined as if you reverse the order of the words in a sequence, you get back the original sequence.

You may assume that there exists a longest consecutive palindromic sequence of words and that it is unique.

Sample input files (`input1.txt`, `input2.txt`, `input3.txt`, `input4.txt`, `input5.txt`) are available for testing. These test-cases are not exhaustive. During grading, there will be many more test-cases, some of which will test for corner-cases.

2.1 Details

Your project should contain the following:

- Be written in standard C++ (no external libraries except ones included in C++, specifically `gcc-4.6.3`).
- A `Makefile` (using `GNU Make`) that calls `gcc` or `g++` to compile your source files to generate an executable named `palindrome`
- Tar and gzip your source files and Makefile for submission into a filename named `<student number>.tar.gz`

- Your program should accept a single command line argument for the input file and **only** output the longest consecutive palindromic sequence of words across all lines (*i.e.* do *not* add other extraneous output like `the output is: here the bottom the here`, this will be considered incorrect output).

An example of how the tester program will run your code, as well as what the output should look like:

```
> tar -xvzf 123456789.tar.gz
> cp *.cpp *.h Makefile > $WORKING_DIR
> cd $WORKING_DIR
> make
> ./palindrome /path/to/input0.txt
here the bottom the here
```

Note that the tester assumes that you only have three types of files `.cpp`, `.h`, and `Makefile` from which your program can be compiled and run from source. `$WORKING_DIR` represents a temporary directory where your program will be extracted, compiled, and run.

Please ensure that your submission can be extracted, compiled and run as shown above as there will be minimal (if any) effort done to fix your submission by the marker if something fails. If your program fails to extract, compile or run, you will receive a grade of **zero** (see marking scheme below).

2.2 Testing

Your program will be extracted, compiled and run on a single core virtual machine running 64-bit Ubuntu Linux with 4GB of RAM using a script similar to the one shown above.

It will be tested against the above input files as well as many test files that will only be available during marking. The test files will test for various things including corner cases and large input sizes. Be sure to make your own test-cases to test your program to ensure that you don't miss any corner cases. The correctness of your program will be determined by comparing your program output to the correct answer.

The efficiency will be judged by the program run-time with respect to all other submissions as well as a reference implementation. If your program fails to finish running within 600 seconds or uses more than 3 GB of RAM for a given execution, it will be considered as a failure to produce the correct output.

3 Deliverables

Exactly two files should be emailed to `briank@eecg.toronto.edu`

1. A PDF report (4 pages max) named `<studentid>.pdf` containing the following:
   (a) Your name, student number, date, and assignment name.
   (b) A brief introduction to the problem.
   (c) High-level description and intuition of your algorithm. Be sure to include descriptions of what algorithms you investigated and why you chose your particular algorithm.
   (d) Description of implementation of your algorithm which should describe the important code, data structures, and optimizations used. Be sure to include descriptions of the different approaches you tried *why* you made a particular choice.
   (e) Big-$\mathcal{O}$ asymptotic analysis of the run-time and memory consumption of your algorithm.
   (f) The output of your program for the five sample input files (`input1.txt`, `input2.txt`, `input3.txt`, `input4.txt`, `input5.txt`).
   (g) A brief conclusion of what you learned.

2. A gzipped tarball named by your student id (`<studentid>.tar.gz`) containing your C++ source code and a Makefile.
   - The tar file should only contain `*.cpp/*.h` files in addition to a single `Makefile`.
   - Your C++ code must be able to compile on `gcc-4.6.3/GNU Make` and run on Linux.

Please request a read receipt when sending the email to ensure you receive confirmation that it has been delivered properly. It is your responsibility to ensure that the email is delivered properly.

3.1 Late Assignments

There will be no late assignments accepted under any circumstances. This includes your computer breaking down, submitting your assignment late by one minute, or any other reason which may or may not be directly under your control. Please plan accordingly.

4 Marking Scheme

The report will be marked out of 100 points. The marking scheme will be a bit unconventional to reflect the emphasis on generating the correct result (because an algorithm isn't useful if it's only correct half the time).

$$grade = (\frac{c}{N})^{1.6}(25 + 50R + 25\lfloor\frac{c}{N}\rfloor P) - S \tag{1}$$

where c is the number of test-cases where your program produced the correct answer, N is the total number of test-cases, R the fractional mark of your report, P is fractional mark of your performance score, and S is a penalty for incorrectly following any instructions (*e.g.* incorrect naming of assignment files, incorrect compression format etc.).

Please notice a few things:

- If your program does not produce correct output for any test cases for any reason, you receive a mark of 0.
- Your mark scales polynomially with the number of correct test-cases.
- You are only eligible to get the performance score if you can pass all the given test cases.
- The report is a significant part of your mark.

Optimize your time accordingly.

5 Useful Resources

- C++ Reference: `http://www.cppreference.com`
- Longest Palindromic Substring:
  `http://leetcode.com/2011/11/longest-palindromic-substring-part-i.html`
- Longest Common Substring problem:
  `http://en.wikipedia.org/wiki/Longest_common_substring_problem`