School of Electrical and Computer Engineering
CptS223: Advanced Data Structures C/C++
Fall 2019
Programming Assignment 1 (PA1)
Warm-Up
Due: 08/30/2019 @ 11:59pm

0. Instructions

All source code must be in C++. You are required to use the Unix environment. Your submission should accompany a report that explains your work. Your code will be tested on EECS servers. Thus, you are strongly encouraged to test (or even write) your code on EECS servers before submitting them. If you do not have an EECS account, you should get one immediately. The EECS IT team will assist you with setting up your EECS account. Look for IT on this page for contact information of the IT personnel:

https://school.eecs.wsu.edu/people/staff/

If you have not logged into EECS Gitlab, you should do so immediately. Information about how to install Git on your computer and how to upload your package (code + report) to the EECS Git can be found under **"Programming Assignments"** section of the course web page. All submission files should be submitted through Git. Note that submissions through any other means for example by emailing your package to the instructor/TA/TAs will be discarded and will NOT be graded. So please follow the above instructions carefully.

You can find the course web page following this link:

http://epsl.eecs.wsu.edu/teaching/

This is an individual programming assignment. No team work is allowed.

The final material for submission should be entirely written by you. If you decide to consult with others or refer materials online, you MUST give due credits to these sources (people, books, webpages, etc.) by listing them in the report that accompanies your submission. Note that no points will be deducted for referencing these sources. However, your discussion/consultation should be limited to the initial design level (if any). Sharing or even showing your source code/assignment solution verbiage to anyone else in the class, or direct reproduction of source code/verbiage from online resources, will all be considered plagiarism, and will therefore be awarded ZERO points and subject to the WSU Academic Dishonesty policy. (Reproducing from the Weiss textbook is an exception to this rule, and such reproduction is encouraged wherever possible.)

Late submission policy: A late penalty of 10% will be assessed for late submissions within the next 24 hours (i.e., until midnight of the next day after the submission deadline). After this one-time late submission, no submissions will be accepted.

1. Introduction

The main purpose of this very first programming assignment is to become familiar with the programming and submission environments new in this course. To this end, this assignment has only small emphasis on algorithm design choices. You will be reporting on your observations about the experiments that will be carried out in this assignment.

2. Project Description

You are asked to implement the code that

- Reads an input file containing integer values;
- Inserts the values into a sorted singly linked list. You can either create a linked list from scratch by yourselves or use the C++ STL libraries. Both approaches are acceptable.
- Reports minimum (referred to as ‘min’), maximum (‘max’), and median (‘med’) of the list
- Reports the time that it takes to
    - insert all the values into the list (‘time_insert’)
    - find minimum of the list after all values are inserted into the list (‘time_min’)
    - find maximum of the list after all the numbers are inserted into the list (‘time_max’)
    - find median of the list after all the numbers are inserted into the list (‘time_med’)

Your program should prompt the user to enter a filename and should print out the output numbers (min, max, med, time_insert, time_min, time_max, time_med) with appropriate messages.

Note that the linked list should contain ‘sorted’ values at any point in time. Thus, you cannot insert all the values into the list first and then try

to sort the numbers. Instead, the list should remain a sorted one after every insertion.

The timing statistics should be in seconds or milliseconds or microseconds (whichever gives the closest precision to measure the actual time of the event). Now, it may so happen that if a particular timed event's time is less than a second, your timer function will show 0 seconds. Obviously this does not mean 0 seconds. It just means you got to measure the time at a lower/finer resolution and use milliseconds or microseconds. If it turns out that the timed event is smaller than a microsecond, then you can consider that time to mean really "0" seconds - i.e., nothing to add to your timer variable.

3. Report

In a separate written document (in Word or PDF), compile sections A through D as follows:

A: Problem statement. In one or two sentences, summarize the goal of the problem.

B: Algorithm design. This project does not have a lot of algorithm design decisions. However, there are few areas that you will need to make a decision about how to compute the output values such as ‘min’, ‘max’, and ‘med’. Within one page, provide a brief description of the main ideas behind your algorithms for finding ‘min’, ‘max’, and ‘med’ within the list. Discuss any potential alternative approaches that could be used to find these values which could have resulted in a poor timing performance.

C: Experimental setup. In this section, you should provide a description of your experiment setup, which includes but is not limited to

- Machine specification.
- How many times did you repeat each experiment before reporting the final timing statistics?

- O/S and environment used during testing: Windows or Unix? Also mention which compiler environment (e.g., gcc). This information will help the TAs determine where to run your programs during grading.

D: Experimental Results & Discussion. In this section, you should report the performance (running time) and outputs based on your observations, and provide justification for your observations. For your testing and reporting, conduct the following two experiments using the two provided input files, namely **'input1.txt'** and **'input2.txt'**.

4. Grading

Assume that the whole assignment is worth 100 points:

CODING (50 pts):

- (15 pts): Is the code implemented in an efficient way? i.e., are there parts in the code that appear redundant, or implemented in ways that can be easily improved? Does the code conform to good coding practices of Objected Oriented programming?
- (15 pts): Does the code compile and run successfully on a couple of test cases?
- (20 pts): Is the code documented well and generally easy to read (with helpful comments and pointers)?

REPORT (50 pts):

- (20 pts): Is the algorithm (for searching min, max, and med) designed efficiently?
- (10 pts): Experimental setup specified.
- (20 pts): Are the justifications/reasons provided to explain the observations analytically sound? Is there a reasonable attempt to explain anomalies (i.e., results that go against analytical expectations), if any?

Obviously to come up with the above evaluation, the TAs are going to both read and run your code.