PROGRAM #2

Fall 2011

Due: October 5, 2011, 5pm on Angel

General Guidelines:

- All source code must be in C++. You can use Windows or Unix environments at your discretion.
- Each program project should be accompanied by a COVER SHEET (see details below). Assignments without cover page will NOT be graded.
- 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 on the cover sheet mentioned below. Note that no points will be deducted for referencing these sources. However, your discussion/consultation should be limited to the initial design level. Sharing or even showing your source code/assignment verbiage to anyone else in the class, or direct reproduction of source code/verbiage from online resources, will all be considered plagiarism, and therefore will 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.)
- Grading will be based on correctness, coding style, implementation efficiency, exception handling, source code documentation, and the written report. Try to also follow the good coding practices discussed in class during the C++ review lecture.
- **How to submit using Angel?**
  All assignments with cover sheet should be zipped or tarred into one archive folder (named after your last name), and attached by email and sent to:
  "***All course faculty***"
  through Angel. This will send out an email to both the instructor and the TA. Submissions are due by 5pm.
  PS: Submissions through any other means or from general email addresses will be discarded and will NOT be graded. So please follow the above instructions carefully.
- Late submission policy: A late penalty of 10% will be assessed for late submissions within the next 24-hour. Note: Submissions will be accepted only through the eLearning webmail. Any other submission outside the eLearning portal will be discarded and not graded. (In the unlikely event of an eLearning server outage on the day of submission, assisgnments can be emailed to the instructor's EECS email account.)

PROBLEM Maximum subsequence sum problem:

For this assignment you will be comparing the performance of the four different algorithms we discussed in class for the maximum subsequence sum problem. Here are the details:

- Implement the four algorithms (maxSubSum1, maxSubSum2, maxSubSum3, maxSubSum4) from the Weiss textbook (pages 52-58). You will need to implement your own max3 function, which is needed to complete the maxSubSum3 code. Foir the maxSubSum4 code, you can either use the Weiss version or the dynamic programming version discussed in class (no need to do both).
- Write a test driver code called *test_driver.cpp* which has a main() function that will allow you to (i) load an arbitrary input array from a text file (to be specified as an argument to the program), (ii) run each algorithm on the input array, and (iii) report their respective running times (the timing should *not* include the initial input load time). Report time in microseconds.
- You should test all the four algorithms on varying input sizes: 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192. For each size, test on 10 different inputs and use the average of these running times to generate the plot for that input size.
- **How to generate inputs:**

  I have generated inputs and have placed in this tarball archive (which you should be able to unzip): inputs.tar

  Alternatively, if you want to generate the inputs yourself, then please follow these instructions: use the program gen_random_arr.c to generate a specified number ("*#samples*") of random arrays, each with positive or negative integers in the range of -9 to +9 for a user-specified input size ("*n*"). You need to compile the code on your machine. (Please read the code to understand what it is doing.) Let us assume *gen_random_arr* be the name of the executable you created. Then the program can be used as follows:

  gen_random_arr {specify array size} {#samples}

  for e.g., "*gen_random_arr 8 10*" will generate 10 different random arrays each with 8 elements. The arrays will be stored and available for use in 10 different files which will be named as: input_8_0.txt, input_8,1.txt, input_8_2.txt, ..., input_8_9.txt.

  You can either use the inputs you generated or the inputs that I have posted above in your tests.
- In each run of your test, you should output:
  - the name of the algorithm (maxSubSum1,maxSubSum2, maxSubSum3, maxSubSum4);
  - the size of the input sequence (n);
  - the final result (maximum subsequence sum); and
  - the total time taken by the algorithm (not including the input loading time).
- **Plots:** Use the above test results and plot the total running times for all 4 algorithms (on Y axis) against each of the input sizes from 8 to 8192 (on X axis). Create a single plot for all the four algorithms so that you are able cross-compare their behavior. Use legends to show which curve is for which algorithm. You must electronic tool such as matlab, gnuplot or excel to create the plot. Hand-drawn plots will not be accepted.

Report:

In a separate document (Word or PDF), compile the following sections:

· *A: Problem statement*. In 1-2 sentences state the goal(s) of this exercise.

- *B: Experimental setup*.
  - Specify the machine architecture (CPU, clock speed, RAM) where all the testing was conducted.
  - Mention whether you used Windows or Unix or Mac OS X for your testing.
  - How many experiments were performed and averaged, to determine each point in your plot?

· *C*: *Experimental Results:* In this section, include the following:

  - The plots from the above test results
  - Are the observations made in the above plots as per your theoretical expectations? If so, why, and if not, why not? Explain.

FINAL CHECKLIST FOR SUBMISSION:

___ Cover sheet

___ A separate folder containing all your source code (including the main function you used for testing)

___ Report

___ Both the above zipped into another folder archive called Program2<YourLastName>.zip