PROGRAM #1

Fall 2011

Due: September 21, 2011, 5pm, submission 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.)

ASSIGNMENT:

The goal of this programming exercise is to identify the performance and implementation tradeoffs between the linked list and array data structures, by implementing the following game.

The *MyJosephus problem* is the following game: N people, numbered 0 to N-1, are sitting in a circle. Starting at person 0, a hot potato is passed. After M passes, the person holding the hot potato is eliminated, the circle closes ranks, and the game continues with the person who was sitting after the eliminated person picking up the hot potato to begin the next round of passing. The game is played until only one person is left, and that last remaining person wins. For example, if M=0 and N=5, players are eliminated in order, and player 4 (i.e., the 5th player) wins; If M=1 and N=5, the order of elimination is 1,3,0,4 before 2 wins.

An animated example for M=2, N=5 is provided here.

Your task is to provide two different implementations for the MyJosephus problem using STL list and vector, respectively. Your program should build upon the following source code:

- Person.h encapsulates each player in the game;
- ListMyJosephus.h provides the required class interface for the list implementation;
- VectorMyJosephus.h provides the required class interface for the vector implementation;
- ElapsedTimeExample.cpp is an example code that shows how to calculate processing time in C++. You can also use other timing functions like *gettimeofday()*.

Using the above source code, implement your own ListMyJosephus.cpp, VectorMyJosephus.cpp, and Person.cpp. Feel free to add more functions as needed for your implementation. The above source code is only given as a template to start with.

For testing and reporting, implement two separate test programs called testListMyJosephus.cpp and testVectorMyJosephus.cpp. These programs should implement the following steps:

1. Instantiate an object of the corresponding MyJosephus class with a user-supplied parameter values for N and M;
2. Play a full game until a winner is found;
3. After each elimination round, output the list of players still left in the game, starting from the lowest player id.
4. At the end of the game, report the elimination sequence and the winner;
5. Report the following timing statistics:
   i) total time for playing the game,
   ii) *average elimination time* which is the average for the time spent between two consecutive eliminations.
   The timing statistics should be in seconds or milliseconds or microseconds (whichever gives the closest precision to measure the actual time of the event).

REPORT:

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

- *A: Problem statement.* In one or two sentences, summarize the goal of the problem.
- *B: Algorithm design*. Within one page, provide a brief description of the *main ideas* behind your algorithms for the two implementations. You can do this in the form of a step-by-step pseudocode (in plain English) along with figures (preferred).
- *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., g++, Visual Studio). This information will help the TAs determine where to run your programs during grading.

- *D: Experimental Results & Discussion.* In this section, you should compare the performance (running time) of the list and vector implementations, and provide justification for your observations.

Results:
For your testing and reporting, conduct the following two sets of experiments separately for each of your two implementations:

- Experiment #1: Keep M fixed at 3, and vary N in powers of two starting from 4,8,16,32,.. up to 1,024. Go even higher if possible, as long as the overall time is under a few minutes.
- Experiment #2: Keep N fixed (at say 512 or 1,024) and vary M in powers of two starting from 2,4,8,... up to the largest power of 2 less than N.

In your report for the results, address the following points:

- Make 4 plots for *each* of your two implementations.
  Plot I) total running time on y-axis vs. N on x-axis;
  Plot II) average elimination time between on y-axis vs. N on x-axis;
  Plot III) total running time on y-axis vs. M on x-axis;
  Plot IV) average elimination time on y-axis vs. M on x-axis;

- You must use some electronic tool (e.g., excel, matlab, gnuplot) to create the plot. Hand-drawn plots will NOT be accepted. The plots should be pasted into the report. *Do not* attach the source for the plot generator.

Discussion:

- Provide a discussion of your results, which includes but is not limited to:
  1. *Which* of the two implementations (list vs. vector) performs the best and under what conditions? Does it depend on the input?
  2. *How* does the running time dependency on the parameter N compare with the dependency on the parameter M?
- Support all your observations made with solid to-the-point justification. For e.g., are the observations consistent with your theoretical expectations or are they in contrary? If so, why? Any theoretical analysis to help understand and explain the observations or discrepancies conceptually will be the best form of justification. This subsection is very important!

COVER SHEET:

Each submission should be accompanied by a cover sheet which is a single page document that contains all the information listed in this sample cover sheet: Word or PDF. Assignments without a cover sheet will NOT be graded.

CHECKLIST FOR SUBMISSION:

___ Cover sheet

___ Zipped folder of the source code

___ Written report

___ All the above three zipped into another folder archive (named after your last name)