EEL 6935: Distributed Computing Spring 2017 Instructor: J. Fortes

Homework 2 Assigned: 1/30/2017 Due: 2/09/2017 @ 11 Pm (use UF E-learning - Canvas)

Multi-threading

Preamble:

In this homework, you are required to write a C/C++/Java program to implement the problem discussed below.

An interesting problem in arithmetic with deep implications to *elliptic curve theory* is the problem of finding perfect squares that are sums of consecutive squares. A classic example is the Pythagorean identity:

$$3^2 + 4^2 = 5^2$$

that reveals that the sum of squares of 3; 4 is itself a square. A more interesting example is Lucas` Square Pyramid:

$$1^2 + 2^2 + \ldots + 24^2 = 70^2$$

In both examples, sums of squares of consecutive integers form the square of another integer.

In this homework, you will be writing a multi-threaded program in either C/C++/Java to build a good solution to this problem. For this assignment, think of workers (threads in C/C++/Java) as solving a range of problems and a master that keeps track of all the problems performing the job assignment. You will have two types of threads, a master thread and a slave/worker thread. The master thread will divide the problem into work units (i.e. sub-problems) and assign it as tasks to the slave threads. The slave threads work on the "work units" and return the result to the master. The master keeps track of the slave threads and aggregates the result from the worker threads and computes if the final sum is a perfect square. You should have answer to question 1 before you start working on the program.

Hint: There are multiple ways to implement this problem as a multi-threaded solution. One of the many is to have the master thread spin-up a slave thread for each of the starting points and each slave thread will individually compute the sum of the squares, validate if the sum is a perfect square and return the starting point to the master, if the sum is a perfect square. The master thread will aggregate all the values from the slave threads and print it on the screen.

Tasks:

Input: The input provided to your program will be two numbers: N and k. The starting numbers for the sequences will be 1, 2, …, N. The overall goal of your program is to find all k consecutive numbers, for each starting number limited by 'N', such that the sum of squares is itself a perfect square.

Output: Print on independent lines, the first number in the sequence for each solution.

Like the previous assignment, you will be submitting a main program called perfsquares.<extension>. This solution you submit may or may not contain multiple classes.

- ***C: gcc perfsquares N k***
- ***C++: g++ perfsquares N k***
- ***Java: java perfsquares N k***

Figure 1

Examples:

Example 1:

```
$>perfsquares  3 2
3
```

This example shows that, with the start point between 1 and 3 and sequence of length 2, the valid sequences are {1,2}, {2,3} and {3,4}. The solution is 3, 4 since $3^2 + 4^2 = 5^2$. Note that you need to print only the starting number.

Example 2:

```
$>perfsquares  40 24
1
```

This example shows that, with the start point between 1 and 40 and sequence of length 24, the solution is 1,2,3…,24 since $1^2 + 2^2 + 3^2 + \ldots + 24^2 = 70^2$.

Questions: - **Do not include the questions in the PDF, only mention the question numbers and provide answers.**

1. How do you determine the size of the work unit? (Size of the work unit here refers to the number of sub-problems that a worker gets in a single request from the master). Show that the size that you chose results in best performance for your implementation.

2. What method(s) do you use to determine if your implementation was indeed running in parallel? Explain in detail and show that your implementation was running in parallel. (Points will be deducted if there is no parallelism). Note: Include the system specification in which you implemented and tested your solution. Include the CPU details (number of cores, number of threads/core), RAM, Operating System, Version of the programming language.

3. Show the result of running your program and the time to complete, for:
   i. N = 1,000,000 and k = 4
   ii. N = 100,000 and k = 8

4. What is the largest problem that you managed to solve with your implementation?

Submission (*Thursday, 02/09/2017, 11 Pm*):

1. Your source code containing the main method must be named perfsquares.<extension of your program>. The names of the other supporting classes, if any, is up-to your choice.

2. Comment every module/class/function in your code and use descriptive variable names.

3. Your program MUST run using the command line arguments shown in the figure 1.

4. There are two files that you will be uploading as part of the homework submission. A tar file and a PDF file. **Do NOT tar the PDF file**.
    a. Upload the following files within a SINGLE tar file named "hw2.tar":
        - The source code of your program
        - A file named "makefile" to compile the program. The TA will type only "make" to compile the program. Other methods are not allowed.
        - A text file named "readme.txt" that precisely specifies the running environment including the operating system, language written, compiler version and any software needed to run your program. It also describes the program structure such as files, classes and significant methods. If your source code is in multiple files, describe briefly the content of each file.
        - All these files should be tarred into a file i.e. the following command should work to create the tar file.

            *$> tar cvf <hw2>.tar **

    b. A PDF document (hw2.pdf) containing answers to the questions and your source code PDF. The source code MUST be pasted towards the end of the PDF. Include all your classes/modules/functions in the PDF file.

        NOTE: The PDF file MUST NOT be tarred. You will have two files to submit. A tar that contains your source code, readme file and a make file, and a PDF file that contains plots, answers to questions, definition of distance for the crescent moon and pasted text of your source code.

5. The tar file and the PDF must be uploaded to Canvas.

Submission Policy:

- Do NOT include binary files. Use the file names as specified above. Incorrect submission formats will lead to a grade reduction.
- All submissions are expected by the deadline specified in the homework assignment. Grade is automatically reduced by **25% for every late day**.
- Make sure to test your submitted code using the tar file. If untar, make, compile or any other command needed to executed your program do NOT work, your homework grade will be **zero.**