Programming project# 2 – Part 2

All the programs should be written, compiled, and executed on the IT Linux servers *unless state otherwise*. You may use your own Debian/Linux virtual machine running on Virtual Box for development and testing. However, you are recommended to test your program on IT Linux servers before you submit your program to ReggieNet (Programming projects to develop Linux kernel modules/code are exceptions).

Programs containing compilation errors will receive failing grades. Those containing run-time errors will incur a substantial penalty. You should make a serious effort to complete all programs on time. Programming assignments are individual. You should complete them with your own effort.

Your half-page write-up in plain text format should be placed in the same directory. You will submit a single "ZIP" file that includes all the subdirectories where your programs and documentation are located. The single ZIP file should contain C/C++ source files, Makefiles, and write-ups in plain text format in the following subdirectories appropriately ./1.

- You might want to use the following Linux/Unix command to create a single zip file:
  % zip –r ***yourlastname_firstName_program2part2*.zip** ./1

To complete this assignment successfully, you will need to read Chapter 3 and Chapter 4 (Thread) in Advanced Linux Programming (NOTE: A copy of the book can be downloaded from the ReggieNet website or directly from the author's website). You might want to try the example code in the two chapters to get better understanding of Pthread APIs.

Programming

[1] (50 points)
Create a subdirectory called "1". Change to the subdirectory

Given a large array of *integers* use C (or C++) AND Pthread APIs to write a parallel program to find the **median and the mode**. The parallel program must meet the following requirements:

- Assume that the entire array is stored initially in a text file. Your program needs to read all integer values from the input text file and store them in one location. Then, the entire array is distributed to the different threads for parallel processing.
- The format of the input text file is as follows:
  - The very first line of the input file begins with % and followed by a space character and a number, which indicates how many integer values are stored in the input file.
  - Adjacent integer values in the input text file is separated by "\n".
  - The following is the content of a sample input text file called inputFile.txt:

```
% 7
412
324
2342
12
5
5
```

4

- DO NOT assume the size of input data. The number of integer elements in the input data can be as large as 100,000,000. **You will need to allocate a memory space *dynamically* (e.g., using malloc() in C or new in C++) to store all integers read from the given input file.**
- The input file name should be provided as a command-line parameter of your program. For example,
  - %mt_medianmode_finder number_of_childthreads inputFile.txt
  - Number_of_childthreads indicates how many child threads to be created for parallel execution. This number can vary from 1 to 4.
- Your program needs to create n child threads and all n threads should use an **exactly same thread function**. As mentioned above, n is given as a command-line argument when you run this program. One of the parameters of the thread function should be used to indicate the range of the entire array to be processed by the given thread.
- Each of the newly created child threads will sort the given subarray using a quicksort routine that can be written by yourself from scratch OR quicksort() function provided in C library.
- IF YOU use other sorting algorithm, you will get 0 points for the whole program automatically!!!
- After each of the child threads has completed sorting its own subarray, the main thread merges the n subarrays into a new array. **This SHOULD BE done by comparing the first elements of the subarrays WITHOUT using any recursive algorithm. More details will be presented in class on Monday, Nov. 1**.
- You should try to minimize the memory usage of your program by limiting the number of memory copies.
- **A sample run of the program:**
  % mt_medianmodeaverage_finder 4 inputFile.txt
  The median is 12.
  The mode is 5. 5 and it appeared 2 times.
  The average is 11.

- How can you check if your multithread program has found the correct median value?
  0) Copy inputFile.txt to inputfileWOheader.txt. Remove the first line in inputFileWOheader.txt
  1) Run "sort –n inputFileWOHeader.txt > sorted.txt" (create a sorted version of the input file)
  2) Run "wc sorted.txt " (It returns line count, word count, and character count.)
  3) Using a text editor, identify the value located at line# that is same as line_count/2 or line_count/2+1)
- How can you check if your multithread program has found the correct mode?
  - Well. One possible approach is to use your MS Excel program.
  - Check https://exceljet.net/excel-functions/excel-mode-function
- **Before you write a multithreaded code, you will need to write a single threaded version of the program first.**
- **I am going to provide one or two sample input text files through ReggieNet.**
- You need to provide a write-up (at least 300 words) describing your source code.
- **(extra credit: 5 points)** Run your multi-threaded program using different number of threads (i.e., 6,5, 4, 3, 2, and 1 threads.). In the writeup, compare the performance between the single-threaded version and the multithreaded version in terms of running time. You may use the following approach to measure the running time:
  ~~% date ; st_medianmodeaverage_finder number_of_threads inputfile.txt; date // you can find the difference between two timestamps~~
  % time st_medianmodeaverage_finder number_of_threads inputfile.txt // you can find the difference between two timestamps

Hints:

https://www.mathsisfun.com/mode.html
If you like, you can use *valgrind* on Linux to check if there is any memory leak.(though it is not mandatory)
How to find the mode or Modal value: To find the mode, or modal value, it is best to put the numbers **in order**. Then **count** how many of each number. A number that appears **most often** is the **mode**.

https://www.mathsisfun.com/median.html
How to find the median value: To find the Median, place the numbers in **value order** and find the **middle**.

Deliverables: a single zip file that contains source code, Makefile and separate written document in plain text format (at least 300 words)

Self-check List:

- **DO NOT rely on source code on Internet including Github!**
- All the values in an input file can be read into a dynamically allocated memory correctly.
- Avoid memory copies as much as possible. DO NOT copy arrays unless it is absolutely necessary!