**Individual Assignment 1 (15%)**

CSE 4600 (Section 01) – Operating Systems – Spring 2022

Submitted to

Department of Computer Science and Engineering

California State University, San Bernardino, California

by

Nathan Bush (007463099)

Due Date: April 22, 2022

*No copy and paste from other colleagues with the same answers and description (particularly in part 1) is allowed. It is required that you carry out the exercises by yourself (with the possibility for collaborating with other colleagues) and provide descriptions (with screenshots wherever necessary) based on your own experience. Copied material from other colleagues will be considered as cheating and dealt with seriously through University academic integrity policies.*

Email: nathan.bush3099@coyote.csusb.edu

Code: <https://github.com/natebush707/CSE_4600_A1.git>

\*\*\* all source .cpp files and screenshots are also included in a .zip file with this report.

1. **(8%) Write a C/C++ program that takes an input (array) from 1 to n (say n = 50) and displays the string representations of those numbers with following conditions**

**If the current number is divisible by 2, then print CSU**

**If the current number is divisible by 5, then print SB**

**If the current number is divisible by both 2 and 5, then print CSUSB**

**If the number is neither divisible by 2 nor 5, then print the number**

**Example:** 1 CSU 3 CSU SB CSU 7 CSU 9 CSUSB 11 CSU 13 CSU SB CSU 17 CSU 19 CSUSB …

**Tasks to do:**

1. (4%) Implement the solution using a single thread (i.e., without using threads and only a main function with any other possible helper function for ease) with all the above conditions implemented in the correct order so that the results are correct. Reason about what can cause the algorithm to print unwanted results (think about the order in which you will write the above conditions). Provide your reasoning about the correct order and the result using single threaded application.

Filename: part1\_1.cpp

Screenshot of code:

Text

Description automatically generated

Screenshot of output:

Graphical user interface, text

Description automatically generated

Report:

The order of determining what to output needs to be specific because of the requirement to output ‘CSUSB’ when the integer is divisible by both 2 and 5. That condition needs to be checked first, or else the branching logic will evaluate both divisible by 2 and/or divisible by 5 to TRUE and proceed accordingly without ever checking the 2 and 5 condition. Once divisible by 2 AND 5 is checked, if it evaluates to FALSE, it does not matter if divisible by 2 or divisible by 5 is checked next since only one of those two conditions can evaluate to TRUE. I used the modulus operator to check conditions. If n mod 5 == 0, then n is divisible by 5. The same is true for n mod 2. As one can see in the output above, all odd integers are displayed, and all even integers are appropriately assigned the string specified in the problem requirements.

1. (4%) Implement a synchronized multithreaded version of CSUSB with four threads. The same instance of CSUSB (array of 1 to 50) will be passed to four different threads:

* Thread 1 should call csu() to check if divisible by 2 then print CSU.
* Thread 2 should call sb() to check if divisible by 5 then print SB.
* Thread 3 should call csusb() to check if divisible by 2 and 5 then output CSUSB.
* Thread 4 should call number() which should only print the numbers.

Submit your code for both parts as separate files (C, Java or CPP) and submit your reasoning on your results (MS word document) that should include brief descriptions of both parts. For multithreaded part, please include your reasoning on the code portion where synchronization is implemented and why. For help, please refer to the lectures and codes where we implemented Dining philosophers, and readers and writers problems using semaphores and pthread\_cond\_wait variables. You are given the choice between semaphores and pthread conditional wait variables whichever seems easier or convenient to you.

**Note:** *Copy and paste of the solutions from other colleagues or online without any reasoning will be automatically marked 0.* While this is an open book assignment and you can search for help online, copying and pasting solutions from online sources with only the variable names changed is strictly prohibited. Please note that the exact copies of your code from either your colleagues or online solutions will automatically be capped to only 50% even if you provided your own explanation. You should clearly cite the references you used for help to avoid automatic 50% deduction and provide the screenshots of your program along with code files. You can use <https://repl.it> to write your program and get the results (screenshots).

Example screenshot of one of the functions:

Text

Description automatically generated

Screenshot of output:

Text

Description automatically generated

Report:

I chose to use pthread conditional wait variables to implement my solution. The thread that checks for the integer being divisible by both 5 and 2 drove the rest of the threads since it needed to make its check first (same reasons as part 1\_1). The above screenshot shows the implementation of this driver thread. At the beginning it waits to acquire the lock until the main function sends a start signal, giving the other threads time to initialize and enter their wait conditions. Once the start signal is received, the thread enters a while loop that checks to see if it needs to output to the console and continue to iterate or send a signal that it has checked the current value so the other threads can execute. If it sends a signal to the other threads, it awaits a signal that the proper value has been outputted before continuing. Once the array is exhausted, the thread releases its lock, signals, the other threads, and returns. Each thread function is similar in that the logic inside is wrapped in a pthread wait condition, before signaling its logic completion and entering another wait. I chose to have each of the ‘lesser’ threads signal each other once they were done with the lock so they could proceed in order. If an output was made from any of the threads, the signal instead went back to the driver thread to move iteration to the next integer in the array. Full source code is included with this report, but was too long for screenshotting here.

1. **(7%) Write a multithreaded program using only Pthreads that uses several threads to multiply two matrices. The multiplication of matrix A with M rows and L columns, and a matrix B with L rows and N columns gives a resulting matrix C with M rows and N columns, and is given by the formula,**

Picture 1

**In matrix multiplication, each element *Cij* is the dot product of the ith row vector of A with the jth column vector of B. The program uses one thread to calculate a dot product.**

A =

Note: For help, please watch the bulb video (Matrix multiplication using threads) in Week 10 folder that explains the above problem in detail and provides partial solution in SDL thread library.

**What to submit?**

Submit your code (C/C++ file) that either asks for matrix input via the command line or has hard coded matrix values as above (i.e. matrices A and B) in the program which should then run multiple threads using pthread library for each row of A and each column of B. Submit a one paragraph explanation of your edited code as asked above with a screenshot of the result (that should precisely display the Result matrix as given above).

Code Snippets:

Text

Description automatically generated

Text

Description automatically generated

Screenshot of output:

A picture containing text

Description automatically generated

Report:

This program creates a thread for each dot product required to calculate the resulting Matrix C. I hardcoded Matrix A and Matrix B from above and used the result to check whether Matrix C was calculating properly. The main program sets up the threads, creates a buffer for passing inputs to the threads, and iterates through each cell of result matrix. On each iteration, a thread is created with logic for calculating the dot product of that cell. I was struggling with passing unique row and column values to each thread and was using global variables to track everything until I saw the provided video which implemented a buffer. After seeing the buffer logic, I modified by code and reduced some of the global variables I was using. One limitation is that my code is immediately joining threads after creating them, which sort of negates the benefit of multithreading. I would have liked to join them after they all are created and working asynchronously, but I was running into thread synchronization issues and mixed results even with mutex locking. However, the requirements of the problem were met.