

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

[Spencer Wallace \(007463307\)](#)

Due Date: [April 24, 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.

- Acknowledged

Email: 007463307@coyote.csusb.edu

***Source code for all problems can be found in the following repo:**
<https://github.com/SpencerDWallace/CSE4600/tree/master/Spencer-Wallace-007463307-Assignment1>

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

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

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 ...

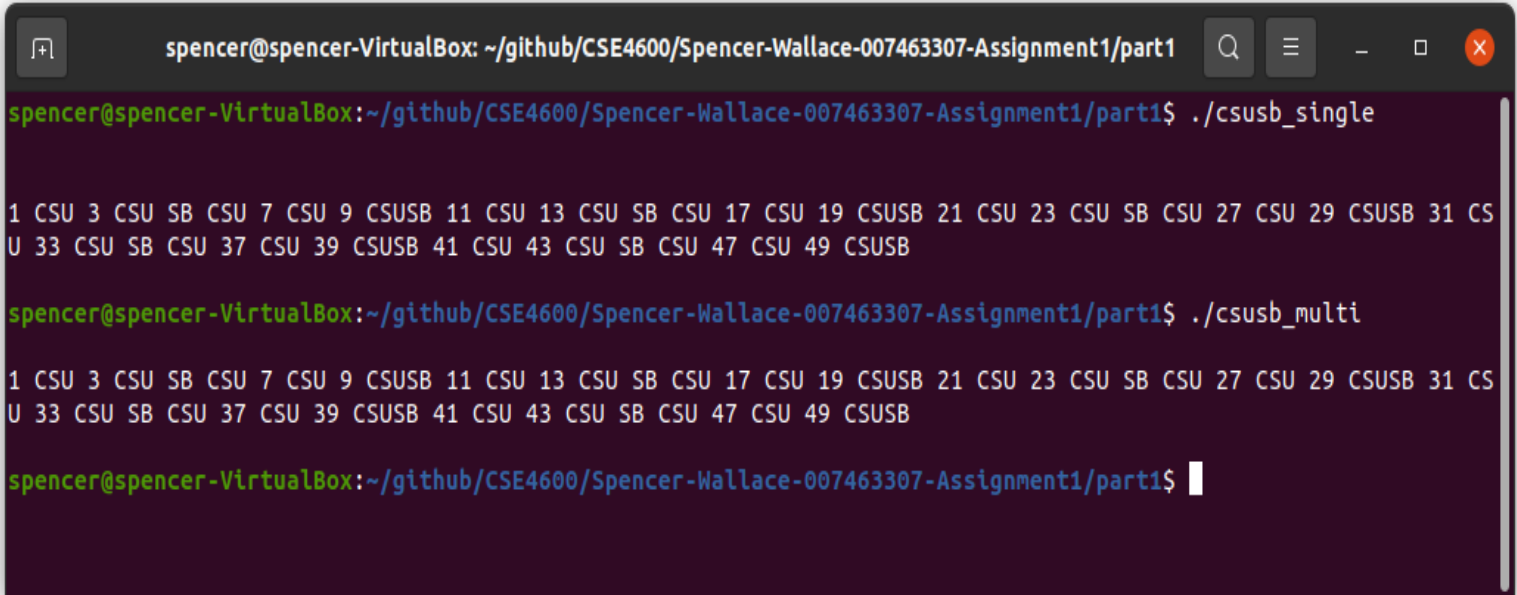
Tasks to do:

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.
2. (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.

Answer on Next Page

Outputs



```
spencer@spencer-VirtualBox: ~/github/CSE4600/Spencer-Wallace-007463307-Assignment1/part1
spencer@spencer-VirtualBox:~/github/CSE4600/Spencer-Wallace-007463307-Assignment1/part1$ ./csusb_single
1 CSU 3 CSU SB CSU 7 CSU 9 CSUSB 11 CSU 13 CSU SB CSU 17 CSU 19 CSUSB 21 CSU 23 CSU SB CSU 27 CSU 29 CSUSB 31 CS
U 33 CSU SB CSU 37 CSU 39 CSUSB 41 CSU 43 CSU SB CSU 47 CSU 49 CSUSB

spencer@spencer-VirtualBox:~/github/CSE4600/Spencer-Wallace-007463307-Assignment1/part1$ ./csusb_multi
1 CSU 3 CSU SB CSU 7 CSU 9 CSUSB 11 CSU 13 CSU SB CSU 17 CSU 19 CSUSB 21 CSU 23 CSU SB CSU 27 CSU 29 CSUSB 31 CS
U 33 CSU SB CSU 37 CSU 39 CSUSB 41 CSU 43 CSU SB CSU 47 CSU 49 CSUSB

spencer@spencer-VirtualBox:~/github/CSE4600/Spencer-Wallace-007463307-Assignment1/part1$
```

Reasoning

For both parts the same functions are used for csu, sb, csusb, and number. There is one difference for the multithreaded programs csu and sb functions – csu checks that it is not divisible by 5 and sb checks that it is not divisible by 2. This was needed so that these statements from these functions would not run for the case that the number is divisible by 10, since 10 is divisible by both 5 and 2 thus the statements from these functions would ordinarily be printed. In order to synchronize the pthreads I used a barrier at the end of each for loop. This was necessary so that each thread would be on the same iteration of the for loop, so that the message would be printed in the correct order (same as single thread/sequential program).

2. (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,

$$C_{ij} = \sum_{k=0}^{L-1} A_{ik} B_{kj} \quad 0 \leq i < M, 0 \leq j < N$$

In matrix multiplication, each element C_{ij} is the dot product of the i^{th} row vector of A with the j^{th} column vector of B. The program uses one thread to calculate a dot product.

$$A = \begin{bmatrix} 5 & 2 & 3 \\ 4 & 5 & 7 \\ 6 & 3 & 7 \\ 1 & 3 & 4 \end{bmatrix}$$
$$B = \begin{bmatrix} 4 & 5 & 6 & 1 \\ 3 & 2 & 3 & 5 \\ 2 & 8 & 7 & 7 \end{bmatrix}$$

$$Result = \begin{bmatrix} 32 & 53 & 57 & 36 \\ 45 & 86 & 88 & 78 \\ 47 & 92 & 94 & 70 \\ 21 & 43 & 43 & 44 \end{bmatrix}$$

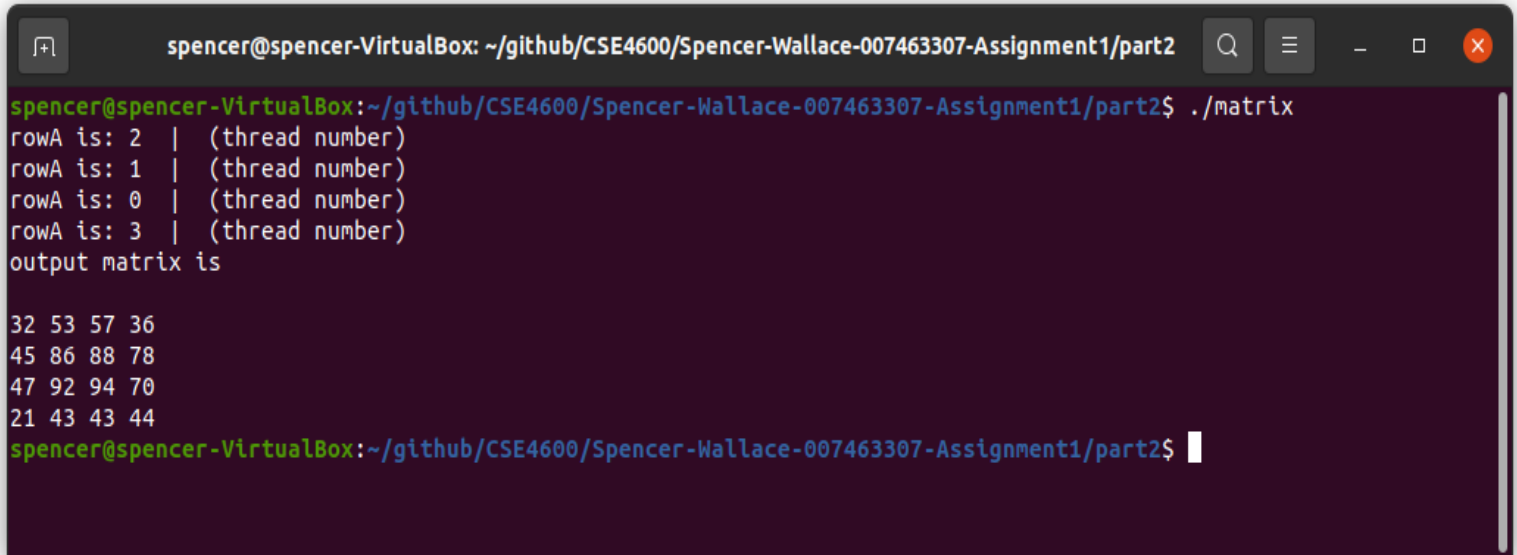
Note: For help, please watch the video via this link ([Matrix multiplication using threads](#)) that explains the above problem in detail and provides partial solution in SDL thread library.

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).

Answer on Next Page

Outputs

A terminal window with a dark purple background. The title bar shows the path ~/github/CSE4600/Spencer-Wallace-007463307-Assignment1/part2. The prompt is spencer@spencer-VirtualBox:~/github/CSE4600/Spencer-Wallace-007463307-Assignment1/part2\$. The command ./matrix has been executed. The output shows four lines of thread identification, followed by the text 'output matrix is', and then a 4x4 matrix of integers.

```
spencer@spencer-VirtualBox:~/github/CSE4600/Spencer-Wallace-007463307-Assignment1/part2$ ./matrix
rowA is: 2 | (thread number)
rowA is: 1 | (thread number)
rowA is: 0 | (thread number)
rowA is: 3 | (thread number)
output matrix is

32 53 57 36
45 86 88 78
47 92 94 70
21 43 43 44
spencer@spencer-VirtualBox:~/github/CSE4600/Spencer-Wallace-007463307-Assignment1/part2$
```

Reasoning

For this program I implemented an algorithm using the standard algorithm for multiplying matrices (as I was taught in linear algebra). I used hard coded matrices for matrix A and B rather than receiving them as input as I felt this would not be very user-friendly/hard to properly validate inputs. To run the algorithm in parallel I created a thread for each row in matrix A, I could have made this more general to work for any sized matrix but in this case I hard coded this to create 4 pthreads. I declared a global array to store the output matrix so that this would not need to be passed to the pthread function, this array is shared between all pthreads and no synchronization is required since each pthread will be outputting to their own specific indexes in this output matrix. To parallelize the algorithm, I passed an index as the argument to the pthread function (0, 1, 2, or 3) which was used to decide what row the pthread would be using to multiply the matrixes as well as the offset for where to begin outputting to the output matrix. This offset was calculated by multiplying the number of columns in A (width of each row) by the index passed to the function. From here the standard algorithm for multiplying matrices was applied.