Roll Num: Name:

CS3523: Operating Systems 2

Programming Lab Exam – Spring 2024

Instructions:

1. **There are two problems mentioned below. Please solve both problems while following the instructions mentioned.**
2. **Using any AI tools, including but not limited to ChatGPT, as a part of your code editor extension is strictly prohibited.**
3. **Please do not copy the code from your friends or elsewhere. Any copying case will result in the student getting F grade and the matter being reported to the Institute Disciplinary Committee.**

Q1: Muli-threaded program for finding Perfect Numbers

Goal:- The objective is to develop a multi-threaded solution to find a list of perfect numbers.

Details:- A perfect number is a positive integer that is equal to the sum of its proper divisors, excluding itself. In other words, a perfect number is a number that is half the sum of all of its positive divisors (including 1 but excluding itself). The smallest perfect number is 6, which has divisors 1, 2, and 3; and 1 + 2 + 3 = 6. Another example is 28, which has divisors 1, 2, 4, 7, and 14; and 1 + 2 + 4 + 7 + 14 = 28.

You need to implement a C/C++ program to find perfect numbers till **N** and list them into a single file. The main thread will read the numbers N and K from an input file. The main thread will create K threads.

The numbers from 1 to N will be partitioned among these K threads so that two threads do not work on the same number. Thus each Thread T_i will be responsible for a set of numbers. For each number in its set, the thread T_i will determine if the number is a perfect number or not.

Input File:- As mentioned above, the input will consist of two parameters, **N** and **K**.

Output File:- For ease of understanding, a log file named OutFile will be created, and each thread will store information about the perfect number in the log file. Suppose Thread T tests a number and finds it to be a perfect number; then, it will print that number and its thread ID. The log file should also contain the total count of the perfect numbers in the given range, N.

A sample log file can be as follows:

```
6: Found by Thread 1
28: Found by Thread 7
.
8128: Found by Thread 2
.
Total Perfect Numbers: 3
```

Submission Guidelines for Q1:- Muli-threaded program

Report Details:- You have to submit a report for this problem. The report should first explain the design of your program while explaining any complications that arose in the course of programming. Specifically, it must explain the logic of partitioning the N numbers among the K threads.

The report should contain the following important results which shows the performance of your program:

1. Time vs Size, N: In this table, show the time taken by your algorithm by varying the values of **N** varying from 10^3 to 10^6 Have K, the number of threads fixed at 8 for all these experiments.

2. Time vs Number of Threads, K: In this table, show the time taken by your algorithm by varying the number of threads from 1 to 16 (in powers of 2 *i.e*, 1, 2, 4, 8, 16). Have N fixed at 10^6 for all these experiments.

Submission Format:- You have to upload: (1) The source code in the following format: perfNum-<RollNo>.c (2) Readme: perfNumReadMe-<RollNo>.txt, which contains the instructions for executing the program. (3) Report: perfNumReport-<RollNo>.pdf, which contains all the details explained above.

Name the zipped document as: LabExam-Q1-<RollNo>.zip. Please follow this naming convention. Otherwise, your assignment will not be graded.

Q2: Paging and File System in xv6

Goal:- The objective is to collect stats on memory consumed by a C program in xv6

Details:- Write a user-level C program **matmul-<rollNo>.c** in xv6 to perform matrix multiplication (C = A * B, where A, B, C are two-dimensional integer arrays of arbitrary size that get their memory from heap dynamically) and gather various stats on memory consumed i.e., page frames allocated for this program on the main memory and details on most and least frequently used page frames and finally the details on the disk blocks allocated for storing the contents of these three matrices on the file system. You can reuse system calls and code base developed in your previous assignments to solve this problem.

When you run matmul-<rollNo> on xv6 shell, it prompts the user with the following:

1. Enter dimensions of matrix A. *Note: you can initialize matrices with arbitrary values after allocating sufficient memory for it from the Heap section of the process. Make sure that your program works even when matrix size goes beyond one page frame.*

2. Enter dimensions of matrix B. *Note: Your program then performs matrix multiplication on A & B and stores the result in C. It creates one file for each of A, B, C named matA.txt, matB.txt, and matC.txt in the file system to store the contents of these matrices in matrix format.*

Your program then prompts the user to choose one of the options given below (use switch statement) to view various stats from this process execution.

Option 1: It should allow the user to specify the name of a matrix and position of an element to display virtual address, virtual page number and corresponding physical address and physical page frame number of that element in the matrix of interest.

Option 2: Prints page table of the process by highlighting page frames allocated for TEXT/Data/Stack/Heap sections of the process. *Note: You may also redirect this output to a file if it's too big.*

Option 3: Prints details on most and least frequently used pages in the Heap section of the process. *Note: Count each read/write memory accesses to determine the frequency of usage of a page in the Heap section. You may also redirect this output to a file if it's too big.*

Option 4: Prints the details on i-node numbers, file sizes, and the disk blocks allocated for storing the contents of these three matrices on the file system. *Note: You may also redirect this output to a file if it's too big.*

Option 5: Exit the process.

Submission Guidelines for Q2:- Paging and File System in xv6

You have to upload: (1) The source code of user-level C program in the following format: matmul-<RollNo>.c, (2) xv6 repository containing all the source files by suitably documenting your modifications with comment boxes, (3) Readme: matmulReadMe-<RollNo>.txt, which contains the instructions for executing the program. (4) Report: matmulReport-<RollNo>.pdf, which contains your solution methodology in terms of new system calls and functions added and list of changes made to various source files in the original xv6 repository for solving this problem **and screenshots of sample outputs.**

Name the zipped document as: LabExam-Q2-<RollNo>.zip. Please follow this naming convention. Otherwise, your assignment will not be graded.