CPSC 5042

Comp Systems Principles II

Homework 2 (100 points)

Due: April 26, Tuesday, 11:00 pm (on Canvas)

NOTE: Please submit your work by April 26 on Canvas. This assignment is to be done individually; you can discuss the questions with your classmates, but you should code your solutions individually.

You can code your solutions in C/C++/Java. Please use one of these languages.

1. Write a multithreaded program (in the file vowels.cpp, or vowels.c, or vowels.java) that counts the number of time each vowel cumulatively appears in 20 different text files. (35 points)

Additional Directions:

- Assume the input files are named "file1.txt", “file2.txt", … , and "file20.txt".
- The program should create 20 threads, each thread should process a different input file.
- Your program must read from the input files from the specified directory.
    - **Do not** assume the input files are in the current directory.
    - The directory must be provided as **command line input**. Prompt the user for this information.
    - File names should be hardcoded in your code, as indicated above.
- Your program must use arrays (or other data structure) and loops to avoid redundant code. Code that replicates something 20 times (such as 20 pthread_create calls) will be penalized.
- If a pthread function returns an error, abort the program with an error message.
- Assuming error-free execution, the only output the program should produce is the cumulative frequency for each of the five vowels (a, e, i, o, and u). **Do NOT** print out the individual counts for each file.
- A vowel should be counted regardless if the vowel is lowercase or uppercase.
- Each input file will contain at least one line. Beyond this, you cannot make any assumptions on how long the input file is.

2. Lazy Dentist Problem: Remember the lazy dentist from HW1… here you will code the solution for this problem: (50 points)

There is a dental clinic with N chairs for waiting patients; the clinic has one doctor. If a patient enters the clinic and there are no free chairs, the patient leaves. If a patient enters the clinic and the dentist is sleeping, the patient wakes up the dentist and consults him. Otherwise, the patient enters the clinic, takes a seat, and waits. If the dentist finishes with one patient and there are waiting patients, the dentist takes the next patient. Otherwise, the dentist goes to sleep in his chair.

A) Below, I am providing the correct solution for this problem. Before proceeding to code, go through the solution and understand it well. You should code the solution outlined below.

B) Write a **semaphore** based solution in C/C++/Java to control the actions of patients and the dentist.

```
// The first two are semaphores are mutexes (only 0 or 1 possible)

Semaphore dentistReady = 0

Semaphore seatCountWriteAccess = 1     // if 1, the number of seats in the waiting room can be
                                       // incremented or decremented

Semaphore patientReady = 0      // the number of patients  currently in the waiting room, ready
to be
                                //served

int numberOfFreeWRSeats = N     // total number of seats in the waiting room
```

```
def Dentist():
while true:                // Run in an infinite loop.
        wait(patientReady)            // Try to acquire a patient - if none is available, go to sleep.
        wait(seatCountWriteAccess)       // Awake - try to get access to modify # of available
      seats,
                                          //otherwise sleep.
        numberOfFreeWRSeats += 1    // One waiting room chair becomes free.
        signal(dentistReady)        // I am ready to consult.
        signal(seatCountWriteAccess)      // Don't need the lock on the chairs anymore.
       // (Talk to patient here.)
```

```
def Customer():
 while true:                // Run in an infinite loop to simulate multiple patients.
        wait(seatCountWriteAccess)        // Try to get access to the waiting room chairs.
        if numberOfFreeWRSeats > 0: // If there are any free seats:
```

```
        numberOfFreeWRSeats -= 1  //   sit down in a chair
        signal(patientReady)        //   notify the dentist, who's waiting until there is a patient
        signal(seatCountWriteAccess)     //   don't need to lock the chairs anymore
        wait(dentistReady)        //   wait until the dentist is ready
        // (Consult dentist here.)
else:                    // otherwise, there are no free seats; tough luck --
        signal(seatCountWriteAccess)     //   but don't forget to release the lock on the seats!
        // (Leave without consulting the dentist.)
```

Testing your code: Add a main method to test your code. Assume N = 3 (number of chairs).

A) Test with one dentist thread and one patient thread.
B) Test with one dentist thread and two patient threads.
C) Test with one dentist thread and three patient threads.
D) Test with one dentist thread and five patient threads.

Additional Directions and Hints:

Add the following print statements in the Dentist thread:

- Before the 1st wait(): “Dentist trying to acquire patient…”
- After 1st wait(), but before 2nd wait(): “Dentist trying to acquire seatCountWriteAccess…”
- After the “numberOfFreeWRSeats += 1” statement: “Incremented free seats to [print updated numberOfFreeWRSeats value here]”
- Before “signal(dentistReady)”: “Dentist ready to consult…”
- Before “ signal(seatCountWriteAccess) ”: “Dentist releasing seatCountWriteAccess…”
- After “ signal(seatCountWriteAccess) ”: “Dentist consulting patient…”

Add the following print statements in each Customer thread; note that you need to include the customer thread number with each of these; the example assumes thread number 3:

- Before “wait(seatCountWriteAccess)”: “Customer 3 trying to acquire seatCountWriteAccess…”

- After “numberOfFreeWRSeats -= 1”: “Customer 3 seated; Remaining chairs=[print updated numberOfFreeWRSeats value here]”
- Before “signal(patientReady)”: “Customer 3 notifying dentist patientReady…”
- Before “signal(seatCountWriteAccess)”: “Customer 3 releasing seatCountWriteAccess…”
- Before “wait(dentistReady) ”: “Customer 3 waiting for dentist…”
- After “wait(dentistReady)”: “Customer 3 consulting dentist…”
- In the else part, before “signal(seatCountWriteAccess) ”: “Customer 3 leaving without consulting; no chairs available…”

Above print statements will help you debug your code and verify its correctness.

3. Prepare a “README.txt” file for your submission. The file should contain the following: (10)

a) Instructions for compiling and executing your program(s). Include an example command line for the same.
b) If your implementation does not work, you should also document the problems in the README file, preferably with your explanation of why it does not work and how you would solve it if you had more time.

4. You should also comment your code well. The best way to go about it is to write comments while coding. (5)

What should you submit?

Upload your submission as a zip file on Canvas. The zip file should contain:

1. All your code files and any other files that might be needed for executing your code.
2. README.txt.