CPSC 351 Project 2 (150 points)

This project can be done individually or in a group of maximum 3 people. For a group of 2 or 3, each of the group members needs to submit. If one group member fails to submit, that person gets 0. Indicate in an additional .txt file, the names and email ids of members in the group. If working individually, indicate in the .txt, your name and email id.

PART A: Computing Fibonacci Numbers With Threads [50 points]

The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8,.... Formally, it can be expressed as:

$fib_0 = 0$

$fib_1 = 1$

$fib_n = fib_{n-1} + fib_{n-2}$

Write a multithreaded C++ program that generates the Fibonacci series using the pthread library. This program should work as follows: The user will enter on the command line the number of Fibonacci numbers that the program will generate. The program will then create a separate thread that will generate the Fibonacci numbers placing the sequence in a data structure that is shared by the threads (a vector is probably the most convenient data structure). Note that the thread function should be iterative when calculating fibonacci (can be recursive, but maybe difficult to implement). When the thread finishes execution, the parent thread will output the sequence generated by the child thread. Because the parent thread cannot begin outputting the Fibonacci sequence until the child thread finishes, this will require having the parent thread wait for the child thread to finish, using the techniques described in lectures pertaining to threads. Note that the vector should only have those many terms as the user wanted.

The name of this program must be fibonacci.cpp

PART B: Printing Words [100 points]

Write a C++ program (using the pthread library) that accepts a phrase of unspecified length on the command line. For example:

```
prompt$: ./vowcon Operating Systems Class at CSUF
```

The main() in this program should read the phrase from the terminal. This phrase can be read into a global variable. This phrase or its parts can be directly accessed from the main() and from the threads. The main() has to create two threads running functions (vow and con). The main()

can also send the phrase or its parts to the threads. The threads should print only the words of the phrase as follows:

The vow thread function should print all the words that start with a vowel

The con thread function should print all the words starting with a consonant.

Note that the main thread (running main()) should not print anything itself, the output should be printed by the threads that it creates. The main() can decide which thread to begin executing first, or it can leave it to the threads to sort this out. **The order of words in the phrase should not be changed in the print**. Your program should work for any phrase of any reasonable length, not just the one given in the example. The output of your program should look similar to the following

```
$ ./vowcon Operating Systems Class at CSUF
vow: Operating
con: Systems
con: Class
vow: at
con: CSUF
```

In this part you are *not allowed* to use synchronization primitives such as semaphores, mutexes, mutex locks such as pthread_mutex_lock and pthread_mutex_unlock and conditional variables such as pthread_cond_wait and pthread_cond_signal etc. from pthreads library for thread coordination. You DO NOT NEED to use sched_yield() to relinquish control of the CPU, but you can use it if you like. You ~~will have to~~ may also investigate some of the other pthread functions available to you for this project, and use them if you like. **Although you do not actually need them**.

The name of this program must be vowcon.cpp

OTHER USEFUL INFORMATION

There are a few systems calls to get and record timing values in user programs available in C/C++. The 'time' system command can also be used as a system call. Other timing operations such as ctime(), gettimeofdday(), ftime(), and time() are system functions and can ~~only~~ also be used in ~~a~~ your programs. CAREFULLY read the man pages for these commands and system calls. **You do not actually need these either.**

On titanium, I have posted many example programs for threads. Take insights from these. The Concurrent threads example can serve as something to start with. Just using basic programming skills and play of loops and if conditions, checking on some variables will help you solve this problem.

FILES TO BE SUBMITTED

1) fibonacci.cpp
2) vowcon.cpp
3) .txt file will name/s and email id/s

Note: Based on the questions in class on Oct 6, and Oct 8, I have added more things in the description.

This is the latest document as of Oct 8. Please read in entirety. All specifications need to be adhered to. At some places, you are also given options, so there you can choose. But at places, where it says for example: output must be printed by so and so, then that must be followed. Mutex locks from pthreads library cannot be used, then that must be followed. And so on.

Anything not specified in the description, there you are free to choose. Of course with everything said, the solution should be your own work, and collaboration is only allowed "WITHIN" the group.

Announcement dated Oct 14 (Project 2 clarification), repeated here:

Part (A) The main( ) function (thread) should create 1 child thread that does the work of generating the fibonacci sequence.

Part (B) The main( ) function (thread) should **ONLY** create 2 child threads : one thread calls vow function, other thread calls con function.

If you have a loop that is creating 2 threads per iteration: when the loop runs n times: 2n threads are created.

But the requirement states: "The main() has to create two threads running functions (vow and con)"

Blurb for your resume

Use your GitHub account as a ready-to-show portfolio of your programming projects to potential employers.