Assignment 1 (Nilanjan)

Given two character strings s1 and s2, use C and pthread to write a parallel program to find out the number of substrings, in string s1, that are exactly the same as string s2. The strings are ended with '\0'. For example, suppose number_substring(s1, s2) implements the function, then number_substring("abcdab", "ab") = 2, number_substring("aaa", "a") = 3, number_substring("abac", "bc") = 0. Suppose the size of s1 and s2 are n1 and n2, respectively, and p threads are used, we assume that n1 mod p = 0, and n2 < n1/p.

Strings s1 and s2 are stored in a file named "strings.txt". String s1 is evenly partitioned for p threads to concurrently search for matching with string s2. After a thread finishes its work and obtains the number of local matching, this local number is added into a global variable showing the total number of matched substrings in string s1. Finally this total number is printed out. The format of the strings.txt is like this (the first string is s1 and the second one is s2):

abcassghbcaj
bca

In the program, use #define to specify number of threads to be created. For example,

#define NUM_THREADS 5

Assignment 2 (Avik)

Use condition variables to implement a producer-consumer algorithm. Here we have two threads: one producer and one consumer. The producer reads characters one by one from a string stored in a file named "string.txt", then writes sequentially these characters into a circular queue. Meanwhile, the consumer reads sequentially from the queue and prints them in the same order. On completion of the running of the program, the string in "string.txt" is printed on the screen. In the program, use #define to specify the size of the queue. For example, #define QUEUE_SIZE 5.

Assignment 3 (Subhas)

Given an array of integers, use C and pthread to write a parallel program to find out the sum of the array and the second maximum. Assume the entire array is stored initially in one location and is distributed to the different threads for parallel processing.

Assignment 4 (Anirban)

Use C and pthread to write a program for a Dental clinic's queuing system that declares an array of integers of size N, where N is the maximum number of queue for the day. The pthread program uses two threads. Whenever there is a new dental appointment, the first thread (the creator) puts the queue numbers in the array, one after the other. The second thread (the remover) removes the queue numbers from the array whenever the dentist has seen the patient. This is done in a FIFO fashion (First In First Out).

The algorithm of the creator is as follows:

- If the array is not full then put a new number in it (the numbers start at 1 and are incremented by one each time, so the creator create queue number 1, 2, 3 etc.)
- sleep for 1 to 10 seconds, randomly
- repeat

- The algorithm of the remover is as follows:
- If the array is not empty then remove its smallest queue number.
- sleep for 1 to 10 seconds, randomly
- repeat

You should use mutex locks to protect things that must be protected. Each thread should print on the screen what it is doing (eg: "number 13 is added into the queue", "number 7 is removed from the queue", etc.). The program should run forever.

Assignment 5 (Sourav)

Write a program using C and pthreads to perform a parallel matrix multiplication routine. The goal is to multiply an MxN matrix called A by an NxP matrix called B and then store the result into the MxP matrix called C. You can design your program in such a manner that each thread does an equal share of the work or you can have the threads run in a loop that computes single rows of the result, C.

Assignment 6 (Avishek)

Write a program using C and pthreads to compute a parallel matrix-vector product. Matrix is distributed by block rows. Vectors are distributed by blocks.

Assignment 7 (Parantapa)

Write a program using C and pthreads to implement a multi-threaded sorted linked list of integers with operations insert, print, member, delete, free list.

Assignment 8 (Ashish)

Given an array of integers, use C and pthread to write a parallel program to sort the array using merge sort. Assume the entire array is stored initially in one location and is distributed to the different threads for parallel processing.

Assignment 9 (Savindra)

Given an array of integers, use C and pthread to write a parallel program to sort the array using insertion sort. Assume the entire array is stored initially in one location and is distributed to the different threads for parallel processing.

Assignment 10 (Apurba)

Given an array of integers, use C and pthread to write a parallel program to find the median. Assume the entire array is stored initially in one location and is distributed to the different threads for parallel processing.

Assignment 11 (Nibedita)

DAXPY stands for "Double precision Alpha X Plus Y." If x and y are n-dimensional arrays of doubles and alpha is a double, then the code for a DAXPY is

```
for (i = 0; i < n; i++)
        y[i] += alpha*x[i];
```

Write a Pthreads program that computes a DAXPY. The main thread should read in n, allocate storage for x and y, and read in x, y and alpha. When the threads have finished computing the DAXPY, the main thread should print the result. You should use a cyclic distribution of the components of x and y, and you should not assume that n is evenly divisible by thread_count. You can make x, y, alpha, and n global variables.

Assignment 12 (Laxman)

Given an array of integers, use C and pthread to write a parallel program to sort the array using quick sort. Assume the entire array is stored initially in one location and is distributed to the different threads for parallel processing.

Assignment 13 (Somabrata)

Given 2 arrays of integers, use C and pthread to write a parallel program to find the common elements. Assume the entire arrays are stored initially in one location and distributed to the different threads for parallel processing.

Assignment 14 (Suvodip)

Estimate pi using the Maclaurin series for arctan(x):

arctan(x) = Sum_n=0 ^infty (-1)^n x^(2n+1)/(2n+1), |x| <= 1

Since arctan(1) = pi/4, we can compute

pi = 4*[1 - 1/3 + 1/5 - 1/7 + 1/9 - . . .]

Run your program as pi_mutex <number of threads> <n>

- pi_mutex is the executable of your code
- n is the number of terms of the Maclaurin series to use
- n should be evenly divisible by the number of threads

Output: The estimate of pi using multiple threads.