CPP-A22: Multithreading <pthread.h>

Assignment 1: Print Fibonacci Series using a Thread

Problem Statement:

Create a thread that prints the first `n` numbers of the Fibonacci series.

Input:

An integer `n = 10`

Expected Output:

```
0 1 1 2 3 5 8 13 21 34
```

Assignment 2: Thread to Check Prime Numbers in a Range

Problem Statement:

Create a thread to print all prime numbers between 1 and `n`.

Input:

An integer `n = 30`

Expected Output:

```
2 3 5 7 11 13 17 19 23 29
```

Assignment 3: Two Threads — One for Even, One for Odd Numbers

Problem Statement:

Create two threads:

- First thread prints even numbers from 1 to 20.
- Second thread prints odd numbers from 1 to 20.

Use `pthread_join()` to ensure both threads complete.

Expected Output:

```
Thread 1: 2 4 6 8 ...
Thread 2: 1 3 5 7 ...
```

(Order may vary based on scheduling.)

Assignment 4: Factorial Calculation Using Thread (Return result from thread)

Problem Statement:

Pass an integer `n` to the thread and calculate `n!` (factorial) inside the thread. Return the result to the main thread.

Input:

`n = 5`

Expected Output:

```
Factorial is: 120
```

Assignment 5: Demonstrate Race Condition Without Mutex

Problem Statement:

Create two threads that increment a shared global variable 100000 times. Do not use any synchronization. Print the final value and observe the issue.

Expected Output:

```
Count is: [less than 200000 due to race condition]
```

Assignment 6: Solve Race Condition Using Mutex

Problem Statement:

Repeat Assignment 5 but this time use a `pthread_mutex_t` lock to prevent race conditions.

Expected Output:

```
Count is: 200000
```

Assignment 7: Sum of Array Elements Using Multiple Threads

Problem Statement:

Divide a large array of size 100 into two halves. Create two threads:

- First sums the first 50 elements.
- Second sums the last 50 elements.

Then combine both results in the main thread.

Expected Output:

```
Sum from thread 1: XXXX
Sum from thread 2: YYYY
Total Sum: ZZZZ
```

Assignment 8: Find Maximum in Array Using Thread

Problem Statement:

Create a thread to find the maximum number from a given array. Pass array as a pointer argument to the thread.

Input Example:

`int arr[] = {5, 2, 9, 6, 4};`

Expected Output:

```
Maximum element is: 9
```

Assignment 9: Use Mutex for Thread-safe Logging

Problem Statement:

Create 3 threads that log messages using `cout`. Use a mutex to synchronize the log printing to prevent interleaved output.

Expected Output:

```
[Thread 1] Logging info
[Thread 2] Logging info
[Thread 3] Logging info
```

Without mutex, logs may mix.

Assignment 10: Parallel Table Generation with Synchronization

Problem Statement:

Create 3 threads:

- Thread 1: Print table of 2
- Thread 2: Print table of 5
- Thread 3: Print table of 10

Use a mutex so only one thread prints its table at a time (clean output).

Expected Output:

```
Table of 2:
2 4 6 8 10 ...

Table of 5:
5 10 15 20 ...

Table of 10:
10 20 30 40 ...
```

Happy Coding!