Experiment 4

Aim:

To write a C program using Pthreads where the main thread creates two worker threads. One thread prints the first 10 prime numbers, and the other prints the first 10 Fibonacci numbers. The main thread waits for both to finish.

Source Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

// Function for the first thread: Primes
void *print_primes(void *arg) {
    printf("Thread 1: First 10 Prime Numbers:\n");
    int count = 0;
    int num = 2;

    while (count < 10) {
        int is_prime = 1;
        // Simple check for prime
        for (int i = 2; i <= num / 2; i++) {
            if (num % i == 0) {
                is_prime = 0;
                break;
            }
        }
        if (is_prime) {
            printf("%d ", num);
            count++;
        }
        num++;
    }
    printf("\nThread 1: Exiting.\n\n");
    pthread_exit(NULL);
}

// Function for the second thread: Fibonacci
```

```c
void *print_fibonacci(void *arg) {
    printf("Thread 2: First 10 Fibonacci Numbers:\n");
    int n = 10;
    int t1 = 0, t2 = 1;
    int nextTerm;

    for (int i = 1; i <= n; ++i) {
        printf("%d ", t1);
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;
    }
    printf("\nThread 2: Exiting.\n\n");
    pthread_exit(NULL);
}

int main() {
    pthread_t thread1, thread2;

    // 1. Create the Prime Number Thread
    if (pthread_create(&thread1, NULL, print_primes, NULL) != 0) {
        perror("Failed to create thread 1");
        return 1;
    }

    // 2. Create the Fibonacci Thread
    if (pthread_create(&thread2, NULL, print_fibonacci, NULL) != 0)
{
        perror("Failed to create thread 2");
        return 1;
    }

    // 3. Main thread waits for both threads to finish
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    // 4. Final message
    printf("I am the main thread\n");

    return 0;
}
```

Compilation & Output:

Command:

```
gcc experiment4.c -o experiment4 -pthread
./experiment4
```

(Note: You must use `-pthread` to link the library)

Output:

```
Thread 1: First 10 Prime Numbers:
2 3 5 7 11 13 17 19 23 29
Thread 1: Exiting.

Thread 2: First 10 Fibonacci Numbers:
0 1 1 2 3 5 8 13 21 34
Thread 2: Exiting.

I am the main thread
```