CS5103: Assignment 9 **Date: 16/10/24**

Example:

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

// The function that will be executed by the thread
void *threadFunction(void *arg) {
    int *num = (int *)arg;
    printf("Hello from the thread! Argument: %d\n", *num);
    pthread_exit(NULL);  // End the thread
}

int main() {
    pthread_t thread;    // Declare a thread variable
    int argument = 10;   // Argument to pass to the thread function
    int result;

    // Create the thread
    result = pthread_create(&thread, NULL, threadFunction, (void *)&argument);

    if (result != 0) {
        printf("Error creating thread: %d\n", result);
        return 1;
    }

    // Wait for the thread to complete
    pthread_join(thread, NULL);

    printf("Thread has finished execution.\n");

    return 0;
}
```

Creating a thread:

1. Include the pthread header: The `pthread.h` header file provides the functions necessary for thread management.
2. Create a thread function: Define the function that the thread will execute. This function must take a `void *` argument and return a `void *`.
3. Create the thread: Use `pthread_create()` to create a new thread.

4. Wait for the thread to complete: Use `pthread_join()` to wait for the thread to finish.

Explanation:

1. `pthread_create()`:
    - Used to create a new thread.
    - It takes four arguments:
        - `pthread_t *thread`: A pointer to a thread identifier.
        - `const pthread_attr_t *attr`: Thread attributes (set to `NULL` for default attributes).
        - `void *(*start_routine)(void *)`: The function to be executed by the thread.
        - `void *arg`: Argument to pass to the thread function.
    - Returns 0 on success or a non-zero error code on failure.
2. `pthread_join()`:
    - Waits for the thread to terminate.
    - The first argument is the thread identifier.
    - The second argument is a pointer to the return value of the thread (can be `NULL` if not needed).
3. `pthread_exit()`:
    - Terminates the calling thread and optionally returns a value.

Question 1:
Create a simple multithreaded program that prints "**Hello from Thread X**" where X is the thread number (e.g., Thread 1, Thread 2, etc.). Use the **pthread library** to create and manage threads.
[5M]

Test Case Description:

Test the program with 5 threads. Each thread should print "Hello from Thread X" where X is the thread number.

Hello from Thread 1

Hello from Thread 2

Hello from Thread 3

Hello from Thread 4

Hello from Thread 5