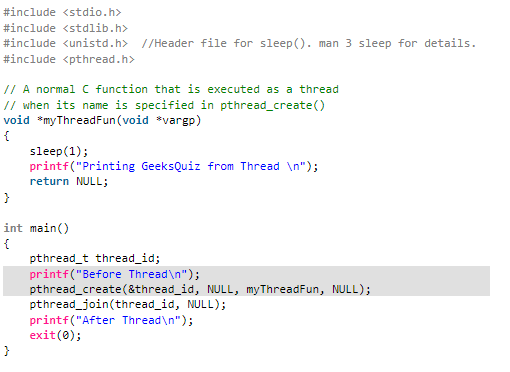
**RIPHAH INTERNATIONAL UNIVERSITY, ISLAMABAD**

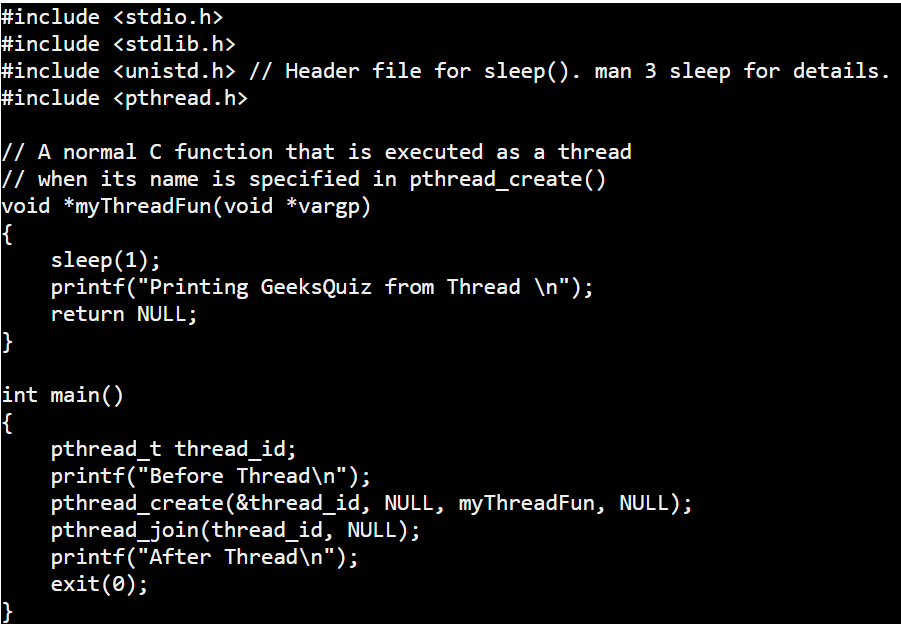
**Lab # 14**

**Alisha Ishrat**

**OS(Lab**)

**Lab Tasks**

**Question: 1**

**Solution:**

**Libraries**

* **stdio.h**: For input and output (e.g., printf).
* **stdlib.h**: For utility functions like exit().
* **unistd.h**: For system calls like sleep().
* **pthread.h**: For thread creation and management (e.g., pthread\_create, pthread\_join).

**myThreadFun Function**

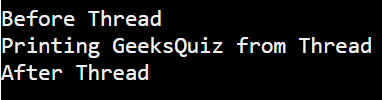
* This is the thread's execution function. It:
  + Pauses execution for 1 second (sleep(1)).
  + Prints "Printing GeeksQuiz from Thread".
  + Returns NULL (since the function has no return value).

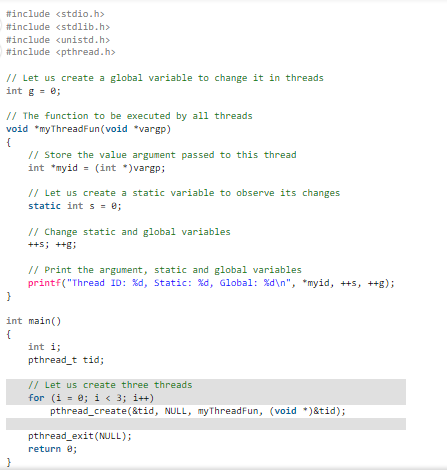
**main Function**

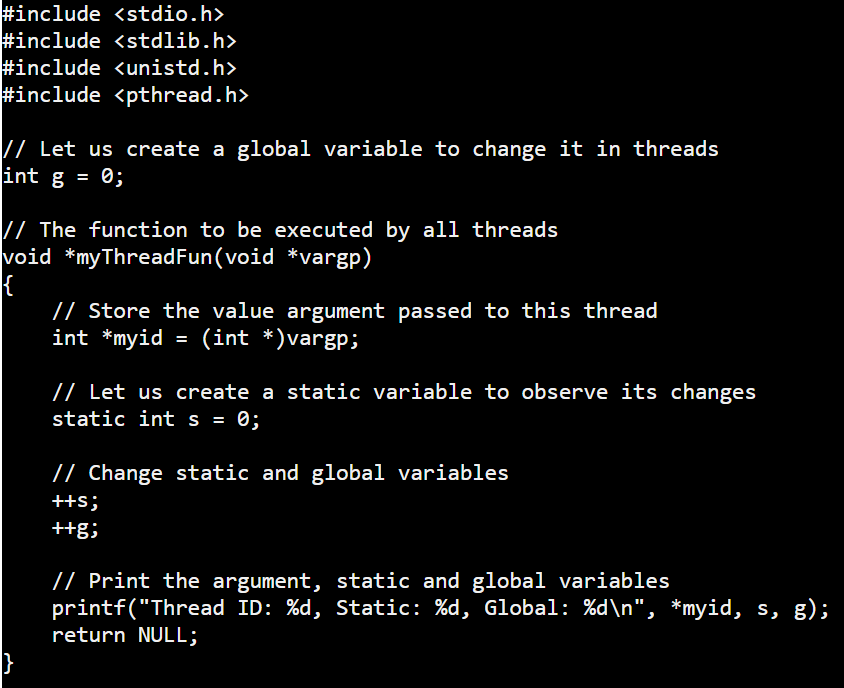
* Creates a thread using pthread\_create.
* Uses pthread\_join to wait for the thread to complete.
* Prints messages before and after the thread is executed.

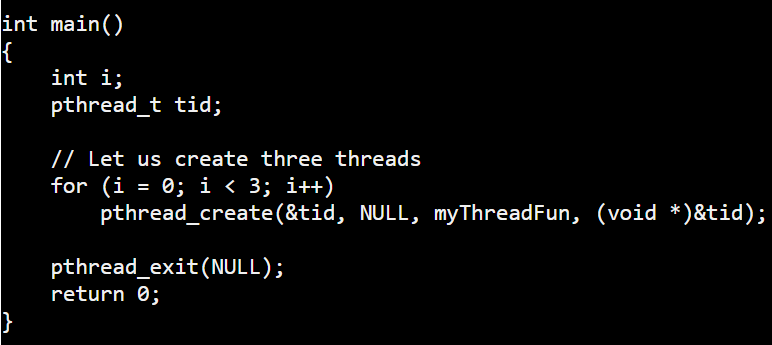
**Output:**

* Output will be:



**Question: 2**

**Solution:**

****

**Global Variable**

* g is a global variable accessible by all threads.

**Static Variable in Thread Function**

* s is static, so it retains its value across multiple calls.

**myThreadFun Function**

* Takes a thread ID as input (vargp).
* Increments the static (s) and global (g) variables.
* Prints the thread's ID along with the updated values of s and g.

**main Function**

* Creates three threads in a loop.
* Each thread calls myThreadFun with its ID.

#### **Step-by-Step Explanation:**

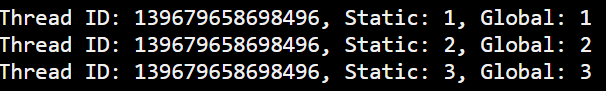
**Thread Initialization**

* Three threads are created using a loop.
* Each thread is passed its index (i) as an argument.

**Shared Variable Behavior**

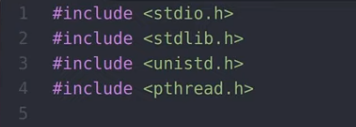
* Static variable s keeps its value across threads because it's shared within the function's scope.
* Global variable g is shared across the entire program and increments with every thread.

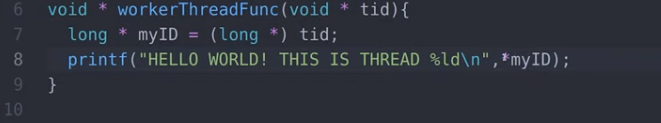
**Output:**

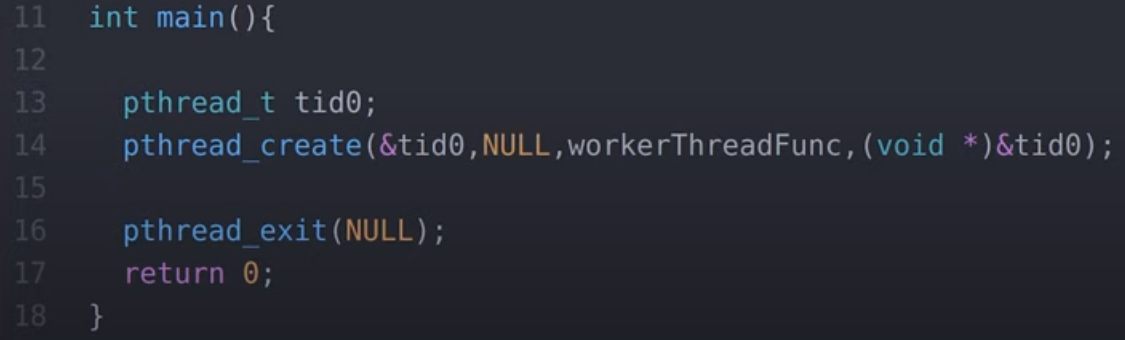


* Each thread prints its ID, the value of s, and the global g.
* Output order may vary due to thread scheduling.

**Question: 3**







**Solution:**

**Creating a New Thread:**

* We create a new thread to do work in parallel with the main program.
* The new thread runs a function (workerThreadFunc()) that prints a message with the thread's ID.

**Key Code Parts**

* pthread\_create(&tid0, NULL, workerThreadFunc, (void\*)&tid0);: This starts a new thread and passes the thread ID to it.
* pthread\_exit(NULL);: Makes the main program wait for the new thread to finish before ending.

**Worker Thread**

The new thread prints "HELLO WORLD! THIS IS THREAD X", where X is the thread’s unique ID.

**Why pthread\_exit()?**

It ensures the main program waits for the thread to finish, preventing it from ending early.

**Output**

****

Where <Thread\_ID> is the unique identifier for each thread. Since you're creating only one thread in this example, it would look like:

****

**Question: 4**

Define posix thread and its working in your own words?

**Solution:**

A **POSIX thread (pthread)** is a unit of execution within a program that follows the POSIX standard for multi-threading, allowing multiple tasks to run concurrently while sharing resources like memory.

**Key Points:**

* **Thread Creation**: pthread\_create() creates a thread, specifying the thread's ID, function to execute, and any arguments.
* **Thread Execution**: Each thread executes the specified function, with its own unique ID, but shares memory with other threads.
* **Synchronization**: Threads use mechanisms like **mutexes** to avoid conflicts when accessing shared resources.
* **Thread Joining**: pthread\_join() makes a thread wait for another thread to finish before continuing.
* **Thread Termination**: A thread can end by returning from its function or calling pthread\_exit().