KASIRALA DURGA-192324073

11.Illustrate the concept of multithreading using a C program.

**Aim:**

To demonstrate the concept of multithreading in C by creating multiple threads that execute concurrently.

**Algorithm:**

1. **Start**.
2. Initialize the program and include the necessary libraries.
3. Define the functions that will be executed by the threads.
4. Create threads using the pthread\_create function.
5. Execute the threads concurrently.
6. Use pthread\_join to wait for threads to finish execution.
7. Print the results from each thread to demonstrate multithreading.
8. **End**.

**Procedure:**

1. Import pthread.h and stdio.h libraries.
2. Define the function for thread execution logic.
3. Use pthread\_create to create multiple threads and pass the function as an argument.
4. Use pthread\_join to ensure main program waits for all threads to finish.
5. Compile and run the program to observe concurrent thread execution.

### Code:

### #include <stdio.h>

### #include <pthread.h>

### #include <unistd.h>

### void \*print\_message(void \*thread\_id) {

### int tid = \*(int \*)thread\_id;

### printf("Thread %d is running\n", tid);

### sleep(1); // Simulate work

### printf("Thread %d has finished\n", tid);

### return NULL;

### }

### int main() {

### pthread\_t threads[3];

### int thread\_ids[3];

### for (int i = 0; i < 3; i++) {

### thread\_ids[i] = i + 1;

### pthread\_create(&threads[i], NULL, print\_message, &thread\_ids[i]);

### }

### for (int i = 0; i < 3; i++) {

### pthread\_join(threads[i], NULL);

### }

### printf("All threads have completed execution.\n");

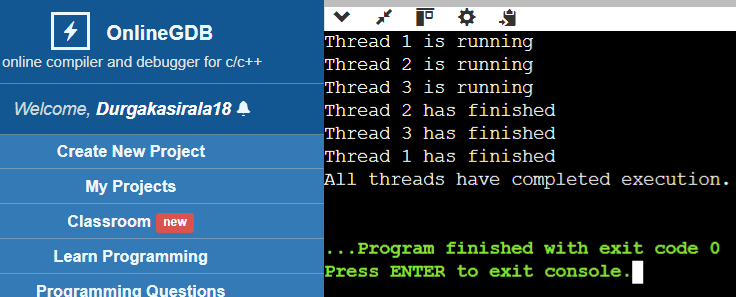
### return 0;

### }

### ****Result****:

When executed, the program creates three threads. Each thread prints its start and end message, demonstrating concurrent execution

**Output:**

****