NATIONAL TEXTILE

UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

SUBMITTED BY:

Farah Naz 23-NTU-CS-1152

SECTION SE: 5th(A)

LAB MANUAL

SUBMITTED TO:

Sir Nasir Mehmood

SUBMISSION DATE:

10-17-2025

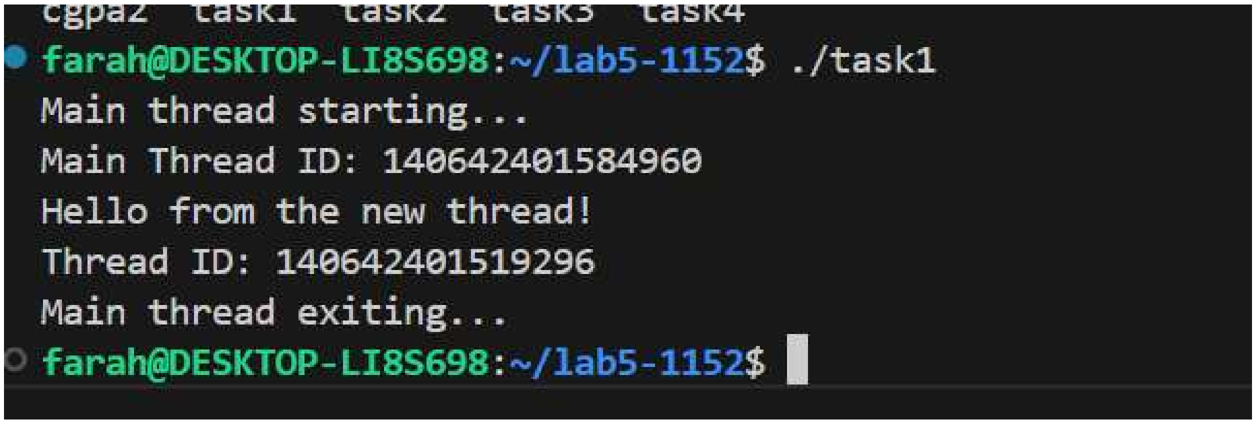
Task 1:

Code:

```c
//CRETAE A THREAD

#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
// Thread function - this will run in the new thread
void* thread_function(void* arg) {
printf("Hello from the new thread!\n");
printf("Thread ID: %lu\n", pthread_self());
return NULL;
}
int main() {
pthread_t thread_id;
printf("Main thread starting...\n");
printf("Main Thread ID: %lu\n", pthread_self());
// Create a new thread
pthread_create(&thread_id, NULL, thread_function, NULL);
// Wait for the thread to finish
pthread_join(thread_id, NULL);
printf("Main thread exiting...\n");
return 0;
}
```

Output:



```
cgpa2  task1  task2  task3  task4
farah@DESKTOP-LI8S698:~/lab5-1152$ ./task1
Main thread starting...
Main Thread ID: 140642401584960
Hello from the new thread!
Thread ID: 140642401519296
Main thread exiting...
farah@DESKTOP-LI8S698:~/lab5-1152$
```

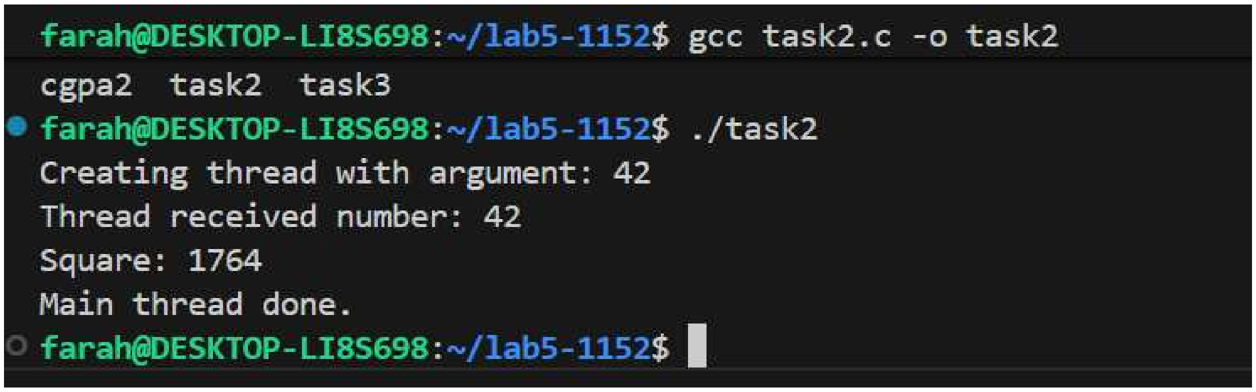
Task 2:

Code:

```c
//PASS ARGUMENTS

#include <stdio.h>
#include <pthread.h>
void* print_number(void* arg) {
// We know that we've passed an integer pointer
int num = *(int*)arg; // Cast void* back to int*
printf("Thread received number: %d\n", num);
printf("Square: %d\n", num * num);
return NULL;
}
int main() {
pthread_t thread_id;
int number = 42;
printf("Creating thread with argument: %d\n", number);
// Pass address of 'number' to thread
pthread_create(&thread_id, NULL, print_number, &number);
pthread_join(thread_id, NULL);
printf("Main thread done.\n");
return 0;
}
```

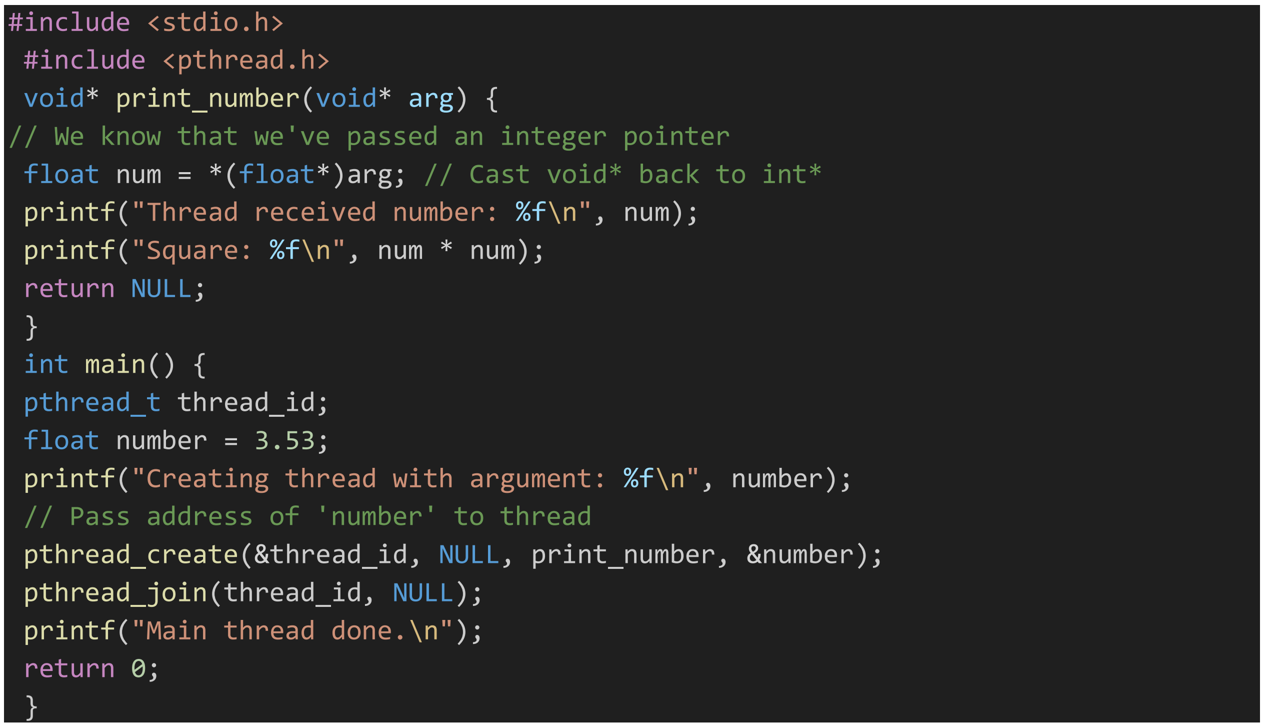
Output:



```
farah@DESKTOP-LI8S698:~/lab5-1152$ gcc task2.c -o task2
cgpa2  task2  task3
farah@DESKTOP-LI8S698:~/lab5-1152$ ./task2
Creating thread with argument: 42
Thread received number: 42
Square: 1764
Main thread done.
farah@DESKTOP-LI8S698:~/lab5-1152$ 
```

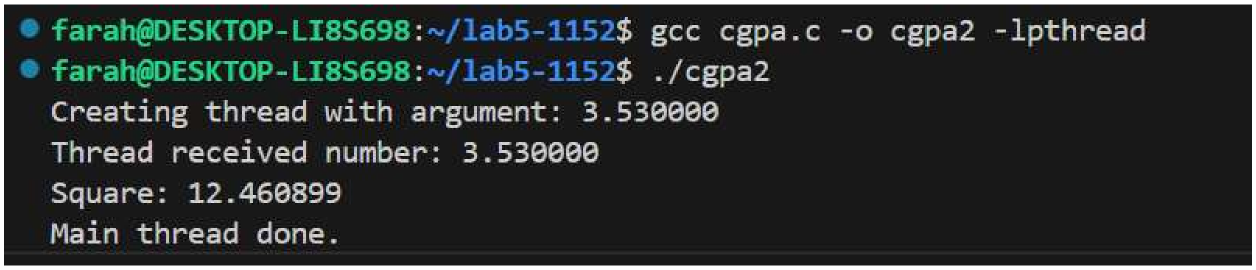
CGPA TASK:

Code:



```c
#include <stdio.h>
 #include <pthread.h>
 void* print_number(void* arg) {
// We know that we've passed an integer pointer
 float num = *(float*)arg; // Cast void* back to int*
 printf("Thread received number: %f\n", num);
 printf("Square: %f\n", num * num);
 return NULL;
 }
 int main() {
 pthread_t thread_id;
 float number = 3.53;
 printf("Creating thread with argument: %f\n", number);
 // Pass address of 'number' to thread
 pthread_create(&thread_id, NULL, print_number, &number);
 pthread_join(thread_id, NULL);
 printf("Main thread done.\n");
 return 0;
 }
```

Output:



```
farah@DESKTOP-LI8S698:~/lab5-1152$ gcc cgpa.c -o cgpa2 -lpthread
farah@DESKTOP-LI8S698:~/lab5-1152$ ./cgpa2
Creating thread with argument: 3.530000
Thread received number: 3.530000
Square: 12.460899
Main thread done.
```

Task 3:

Code:

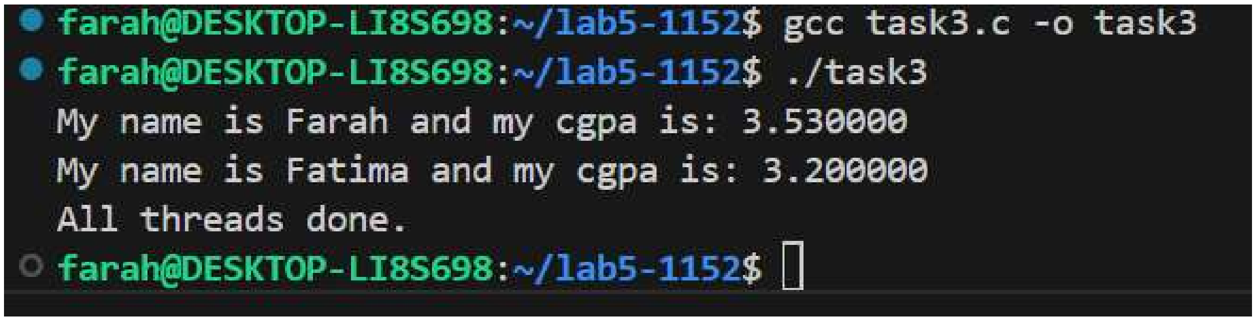
```c
//pass mixed

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

 typedef struct {
 char* name;
 float cgpa;
 } ThreadData;

 void* printData(void* arg) {
 ThreadData* data = (ThreadData*)arg;
 printf("My name is %s and my cgpa is: %f\n", data->name, data->cgpa);
 return NULL;
 }
 int main() {
 pthread_t t1, t2;
 ThreadData data1 = {"Farah", 3.53};
 ThreadData data2 = { "Fatima", 3.2};
 pthread_create(&t1, NULL, printData, &data1);
 pthread_create(&t2, NULL, printData, &data2);
 pthread_join(t1, NULL);
 pthread_join(t2, NULL);
 printf("All threads done.\n");
 return 0;
 }
```

Output:



```
farah@DESKTOP-LI8S698:~/lab5-1152$ gcc task3.c -o task3
farah@DESKTOP-LI8S698:~/lab5-1152$ ./task3
My name is Farah and my cgpa is: 3.530000
My name is Fatima and my cgpa is: 3.200000
All threads done.
farah@DESKTOP-LI8S698:~/lab5-1152$ 
```

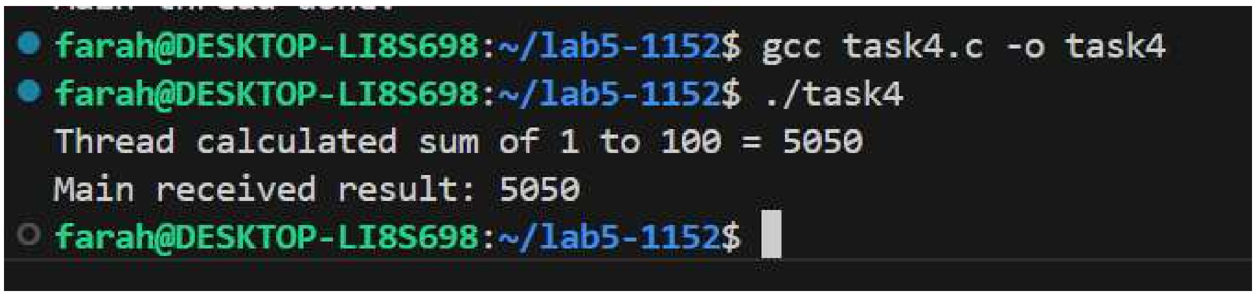
Task 4:

Code:

```
//Return values from a thread

#include <stdio.h>
 #include <pthread.h>
 #include <stdlib.h>
 void* calculate_sum(void* arg) {
 int n = *(int*)arg;
 int* result = malloc(sizeof(int)); // Allocate memory for result
 *result = 0;
 for (int i = 1; i <= n; i++) {
 *result += i;
 }
 printf("Thread calculated sum of 1 to %d = %d\n", n, *result);
 return (void*)result; // Return the result
 }
 int main() {
 pthread_t thread_id;
 int n = 100;
 void* sum;
 pthread_create(&thread_id, NULL, calculate_sum, &n);
 // Get the return value from thread
 pthread_join(thread_id, &sum);
 printf("Main received result: %d\n", *(int*)sum);
 free(sum); // Don't forget to free allocated memory
 return 0;
 }
```

Output:

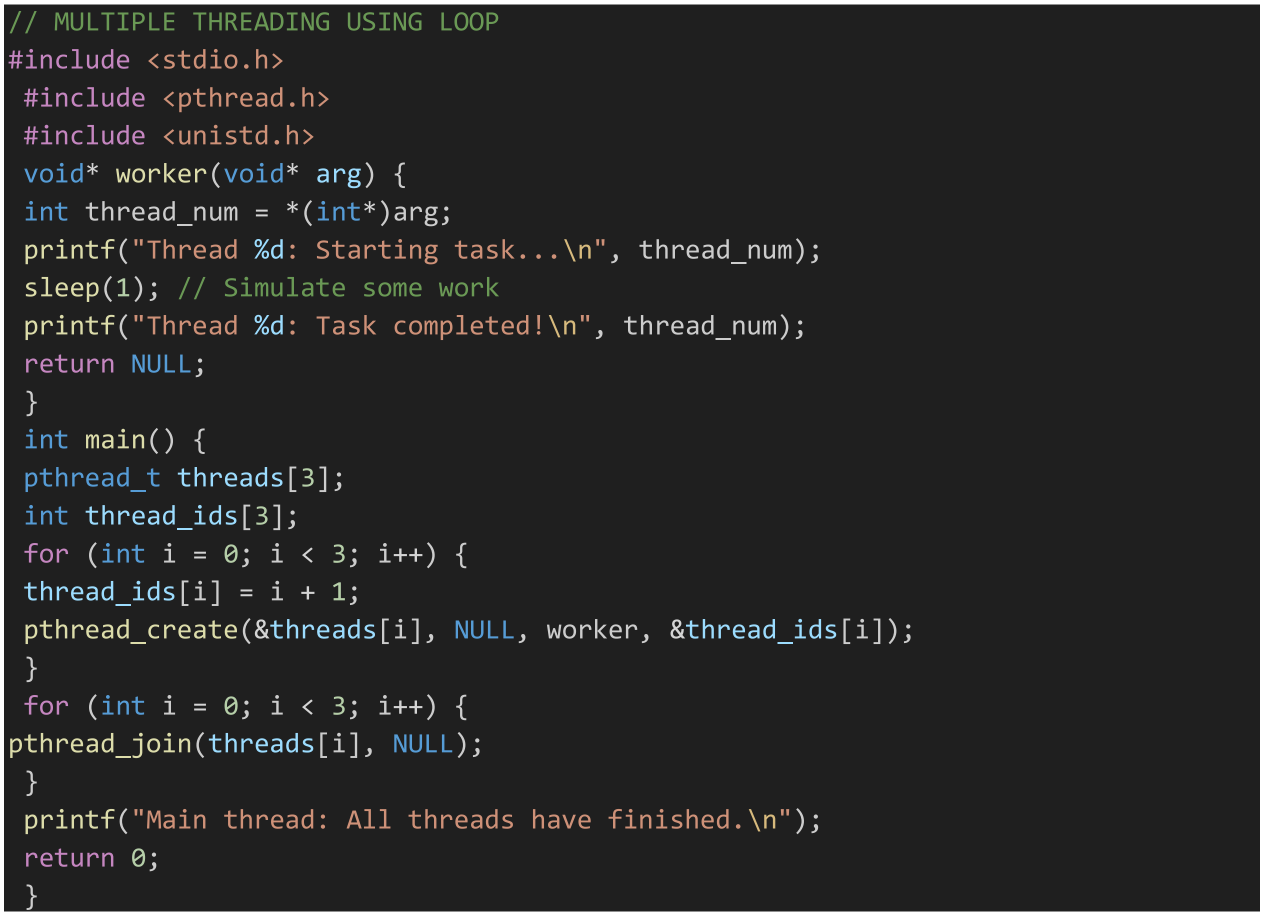


```
farah@DESKTOP-LI8S698:~/lab5-1152$ gcc task4.c -o task4
farah@DESKTOP-LI8S698:~/lab5-1152$ ./task4
Thread calculated sum of 1 to 100 = 5050
Main received result: 5050
farah@DESKTOP-LI8S698:~/lab5-1152$
```

Multi Threading :

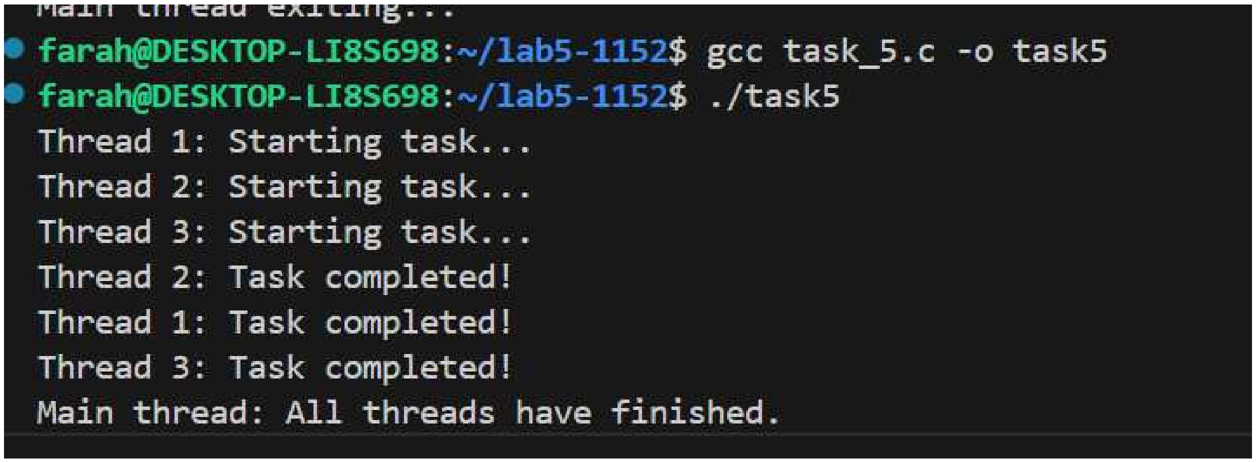
Task 5:

Code:



```
// MULTIPLE THREADING USING LOOP
#include <stdio.h>
 #include <pthread.h>
 #include <unistd.h>
 void* worker(void* arg) {
 int thread_num = *(int*)arg;
 printf("Thread %d: Starting task...\n", thread_num);
 sleep(1); // Simulate some work
 printf("Thread %d: Task completed!\n", thread_num);
 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, worker, &thread_ids[i]);
 }
 for (int i = 0; i < 3; i++) {
pthread_join(threads[i], NULL);
 }
 printf("Main thread: All threads have finished.\n");
 return 0;
 }
```

Output:

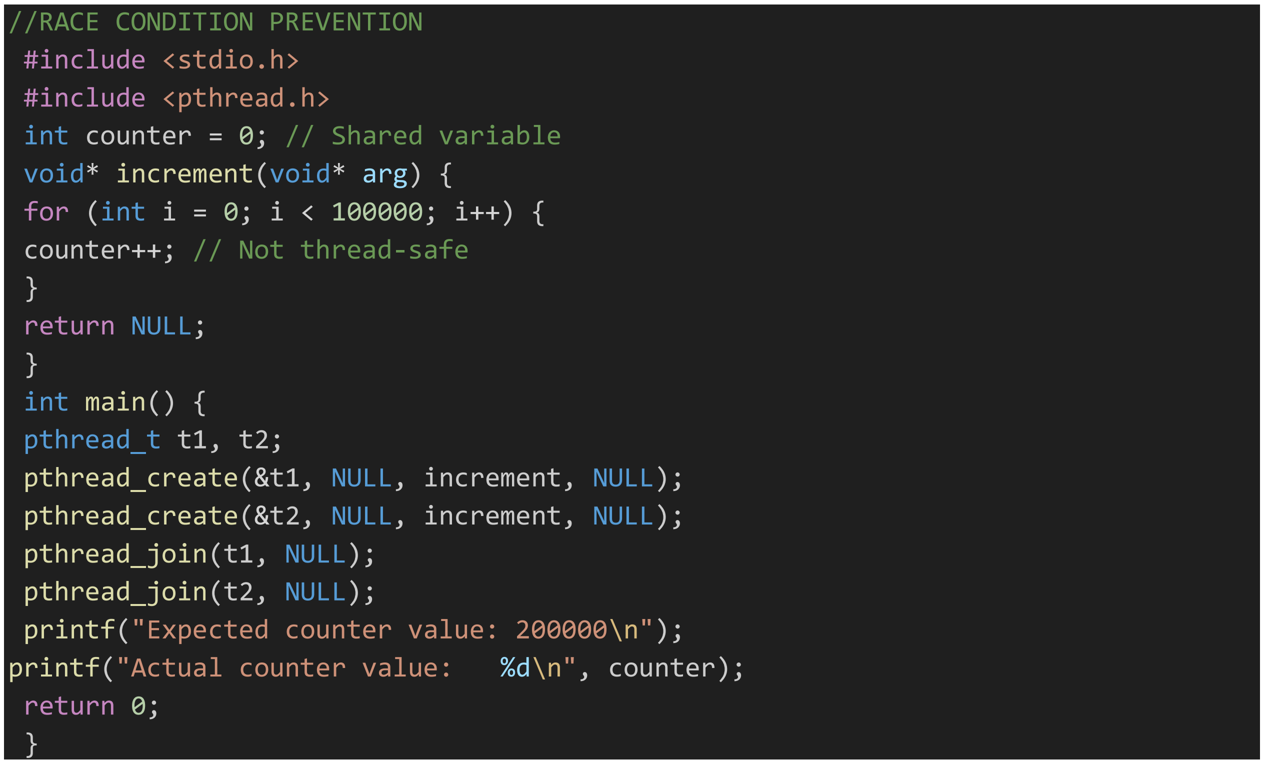


```
  Main thread exiting...
farah@DESKTOP-LI8S698:~/lab5-1152$ gcc task_5.c -o task5
farah@DESKTOP-LI8S698:~/lab5-1152$ ./task5
  Thread 1: Starting task...
  Thread 2: Starting task...
  Thread 3: Starting task...
  Thread 2: Task completed!
  Thread 1: Task completed!
  Thread 3: Task completed!
  Main thread: All threads have finished.
```

Race Condition:

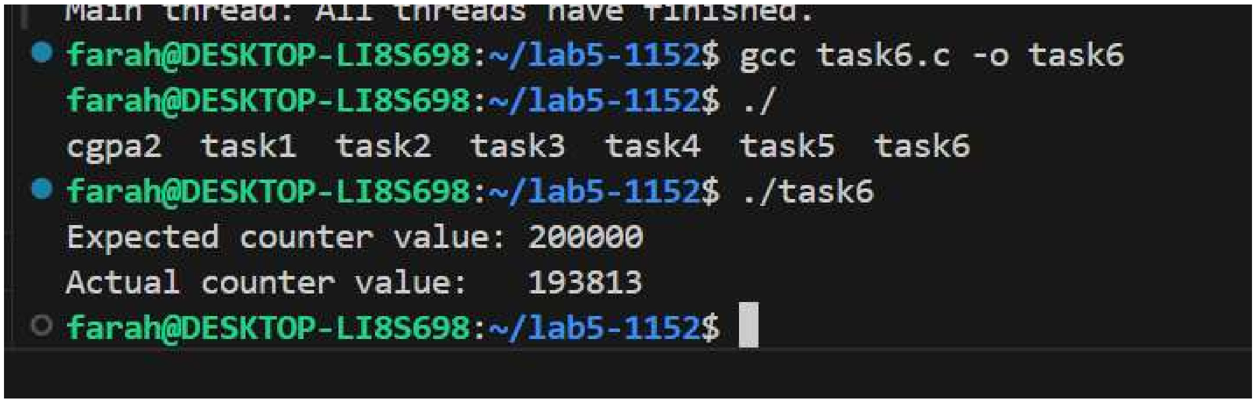
Task 6:

Code:



```c
//RACE CONDITION PREVENTION
 #include <stdio.h>
 #include <pthread.h>
 int counter = 0; // Shared variable
 void* increment(void* arg) {
 for (int i = 0; i < 100000; i++) {
 counter++; // Not thread-safe
 }
 return NULL;
 }
 int main() {
 pthread_t t1, t2;
 pthread_create(&t1, NULL, increment, NULL);
 pthread_create(&t2, NULL, increment, NULL);
 pthread_join(t1, NULL);
 pthread_join(t2, NULL);
 printf("Expected counter value: 200000\n");
printf("Actual counter value:   %d\n", counter);
 return 0;
 }
```

Output:



```
Main thread: All threads have finished.
farah@DESKTOP-LI8S698:~/lab5-1152$ gcc task6.c -o task6
farah@DESKTOP-LI8S698:~/lab5-1152$ ./
cgpa2  task1  task2  task3  task4  task5  task6
farah@DESKTOP-LI8S698:~/lab5-1152$ ./task6
Expected counter value: 200000
Actual counter value:   193813
farah@DESKTOP-LI8S698:~/lab5-1152$ 
```