National Textile University

Department of Computer Science

Subject:

Opreating System

Submitted to:

Sir Nasir

Submitted by:

Ahmad Fawad

Reg number:

1129

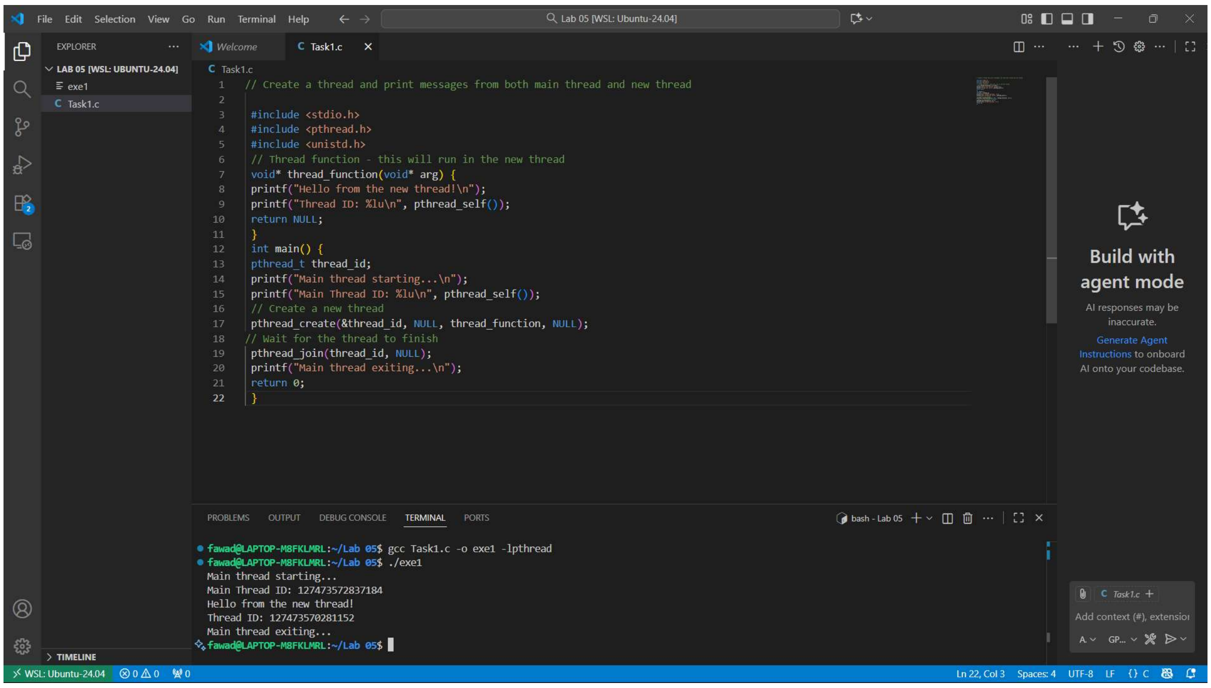
Lab no. :

05

Semester:

5th

Task 1:

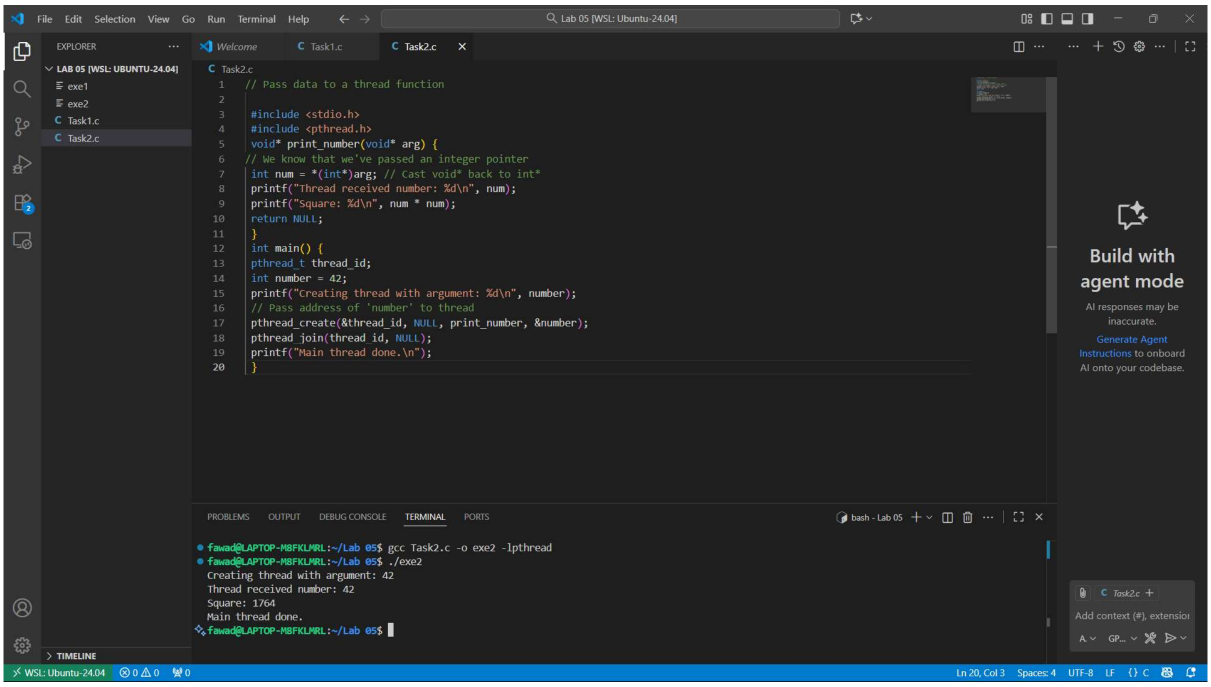


```c
// Create a thread and print messages from both main thread and new thread

 #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;
 }
```

```
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ gcc Task1.c -o exe1 -lpthread
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ ./exe1
Main thread starting...
Main Thread ID: 127473572837184
Hello from the new thread!
Thread ID: 127473570281152
Main thread exiting...
fawad@LAPTOP-M8FKLMRL:~/Lab 05$
```

Task2:

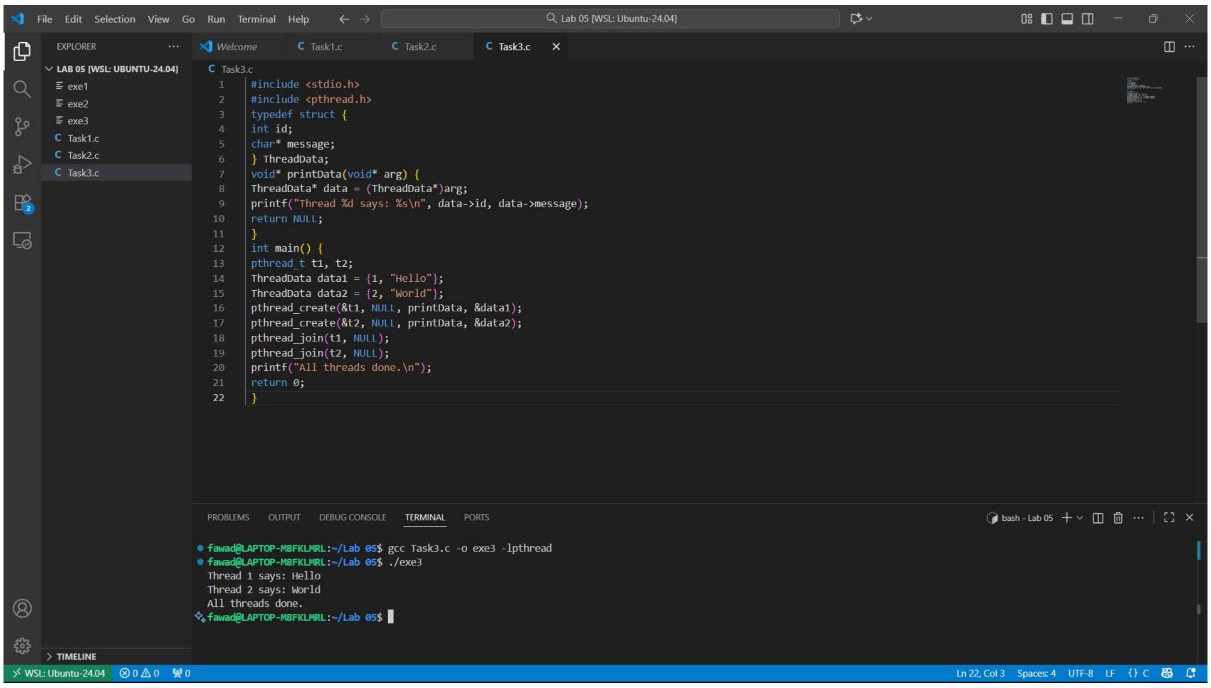


```c
// Pass data to a thread function

 #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");
 }
```

```
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ gcc Task2.c -o exe2 -lpthread
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ ./exe2
Creating thread with argument: 42
Thread received number: 42
Square: 1764
Main thread done.
fawad@LAPTOP-M8FKLMRL:~/Lab 05$
```

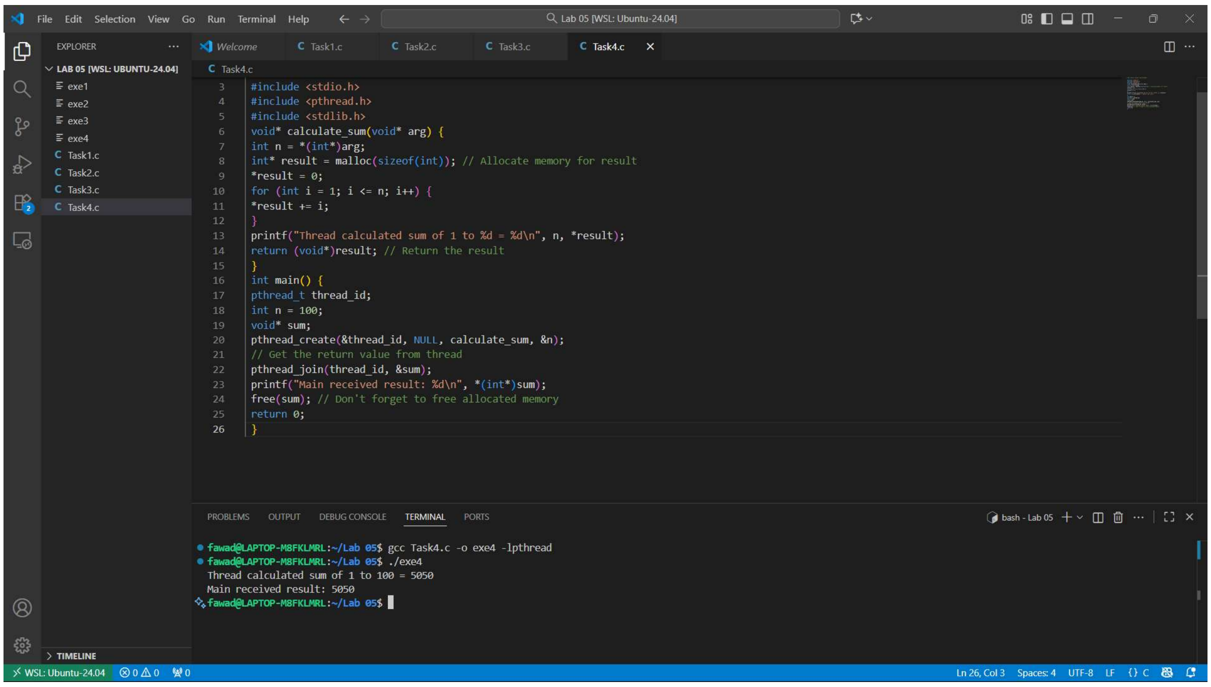
Task3:



```c
#include <stdio.h>
#include <pthread.h>
typedef struct {
int id;
char* message;
} ThreadData;
void* printData(void* arg) {
ThreadData* data = (ThreadData*)arg;
printf("Thread %d says: %s\n", data->id, data->message);
return NULL;
}
int main() {
pthread_t t1, t2;
ThreadData data1 = {1, "Hello"};
ThreadData data2 = {2, "World"};
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;
}
```

```
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ gcc Task3.c -o exe3 -lpthread
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ ./exe3
Thread 1 says: Hello
Thread 2 says: World
All threads done.
fawad@LAPTOP-M8FKLMRL:~/Lab 05$
```

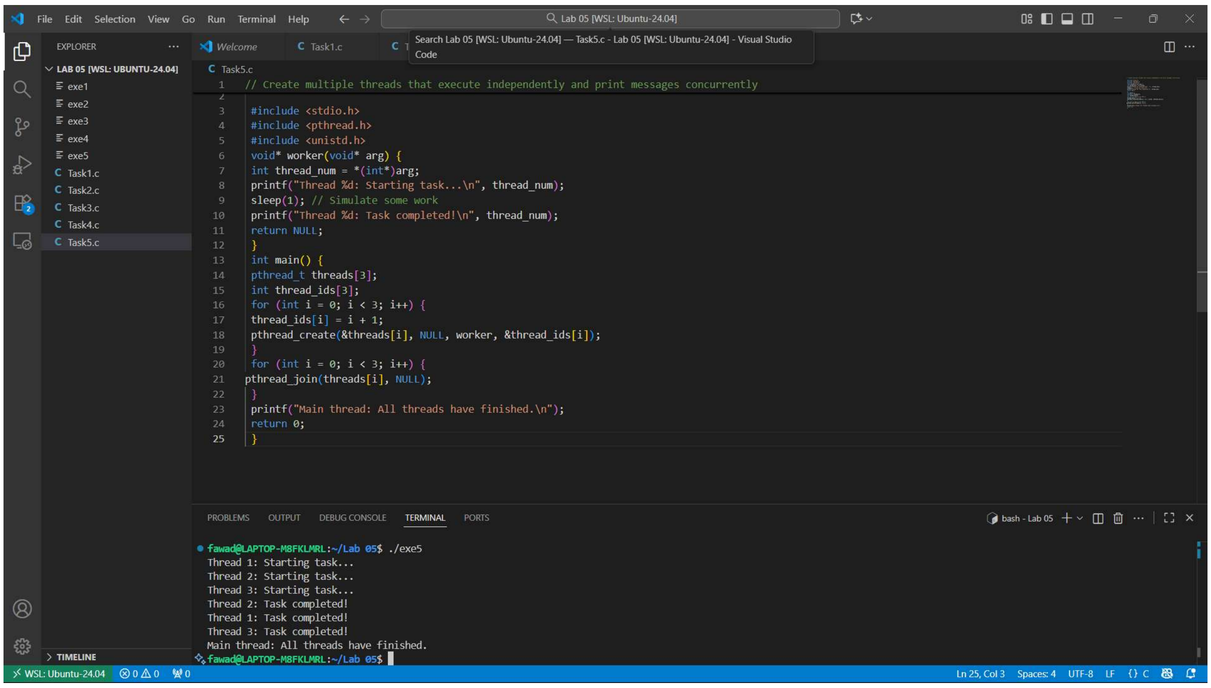
Task 4:



```c
#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;
}
```

```
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ gcc Task4.c -o exe4 -lpthread
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ ./exe4
Thread calculated sum of 1 to 100 = 5050
Main received result: 5050
fawad@LAPTOP-M8FKLMRL:~/Lab 05$
```

Task5:

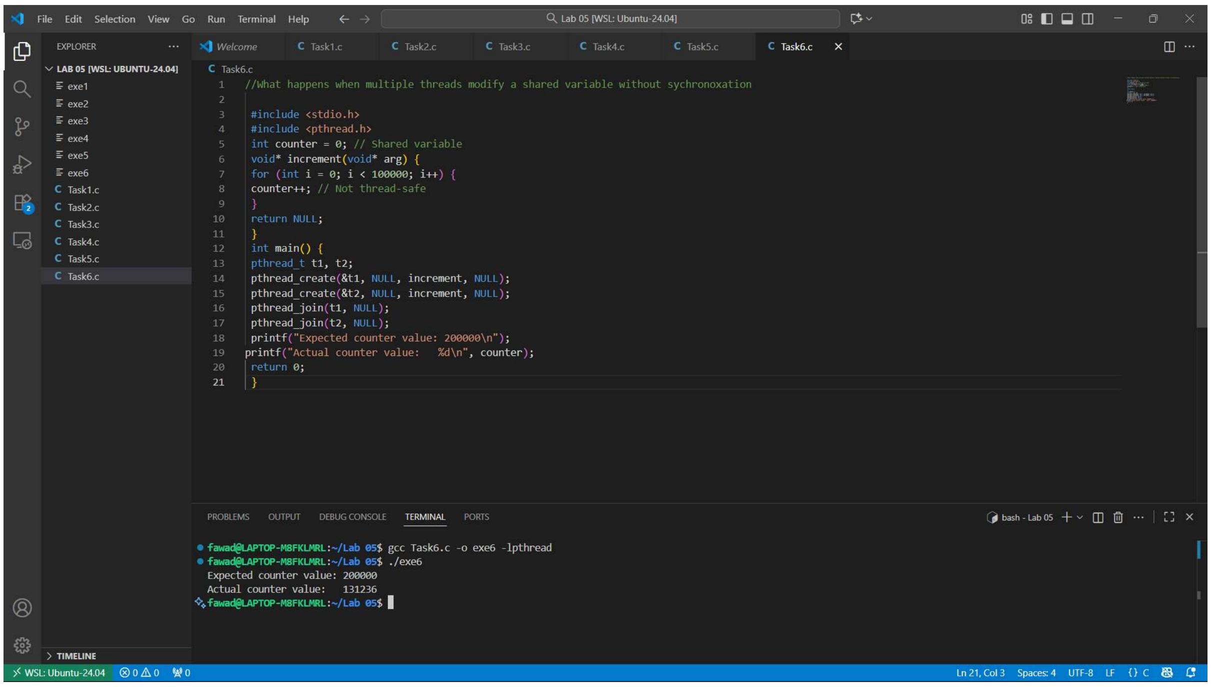


```c
// Create multiple threads that execute independently and print messages concurrently

 #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;
 }
```

```
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ ./exe5
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.
fawad@LAPTOP-M8FKLMRL:~/Lab 05$
```

Task6:



```c
//What happens when multiple threads modify a shared variable without sychronoxation

 #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;
 }
```

```
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ gcc Task6.c -o exe6 -lpthread
fawad@LAPTOP-M8FKLMRL:~/Lab 05$ ./exe6
Expected counter value: 200000
Actual counter value:   131236
fawad@LAPTOP-M8FKLMRL:~/Lab 05$
```