National Textile University

Department of Computer Science

Subject:

Operating system

Submitted To:

Sir Nasir

Submitted By:

Noor ul Ain

Registration No:

23-NTU-CS-1221

Lab No:

4 -Home task

Semester:

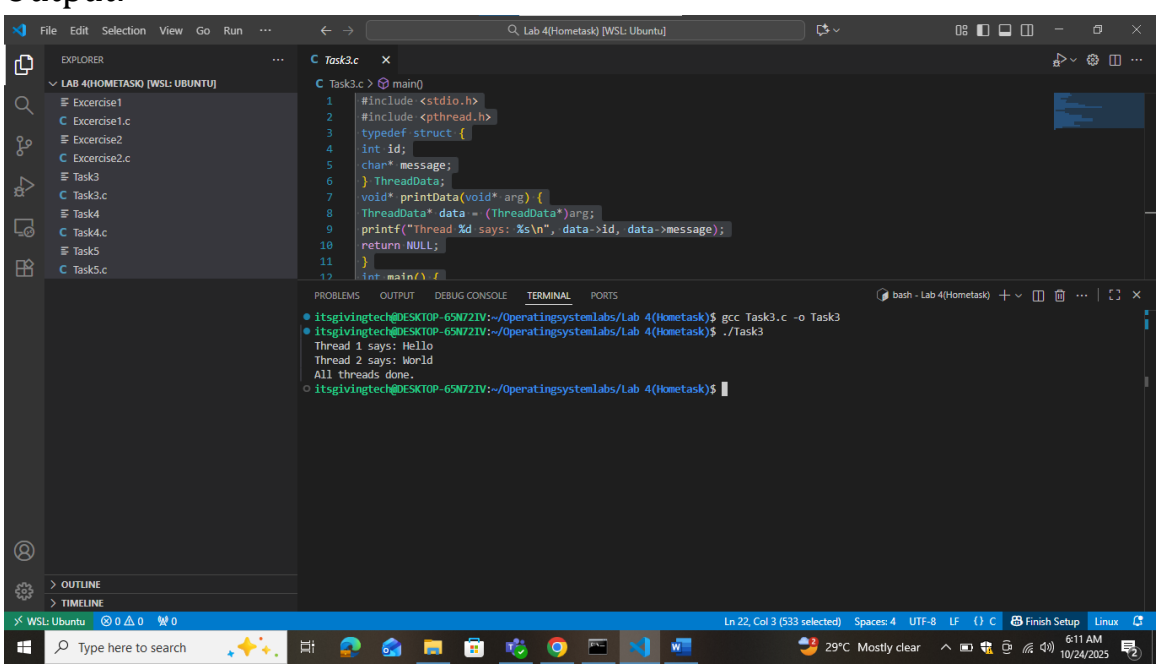
5

Task 3:

Program 3: Passing Multiple Data

```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;
}
```

Output:



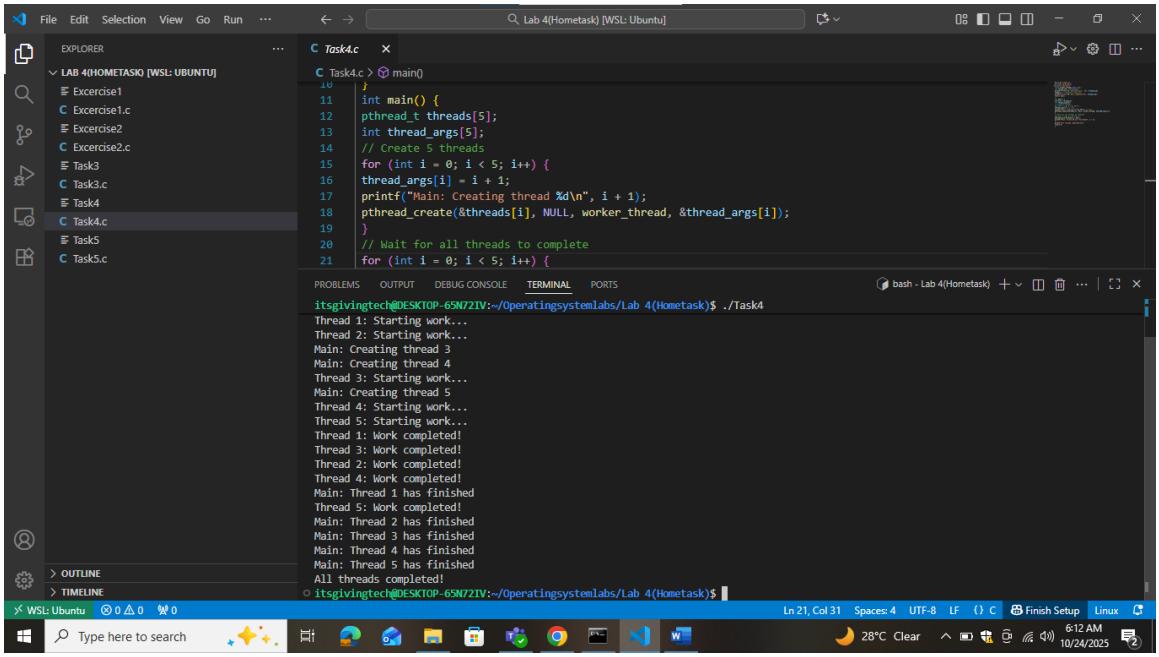
Task 4:

Program 4: Multiple Threads

Code:

```
 #include <stdio.h>
 #include <pthread.h>
#include <unistd.h>
 void* worker_thread(void* arg) {
 int thread_num = *(int*)arg;
 printf("Thread %d: Starting work...\n", thread_num);
 sleep(1);
 printf("Thread %d: Work completed!\n", thread_num);
 return NULL;
 }
 int main() {
 pthread_t threads[5];
 int thread_args[5];
 // Create 5 threads
 for (int i = 0; i < 5; i++) {
 thread_args[i] = i + 1;
 printf("Main: Creating thread %d\n", i + 1);
 pthread_create(&threads[i], NULL, worker_thread, &thread_args[i]);
 }

 for (int i = 0; i < 5; i++) {
 pthread_join(threads[i], NULL);
 printf("Main: Thread %d has finished\n", i + 1);
 }
 printf("All threads completed!\n");
 return 0;
 }
```

Output:



Task 5:

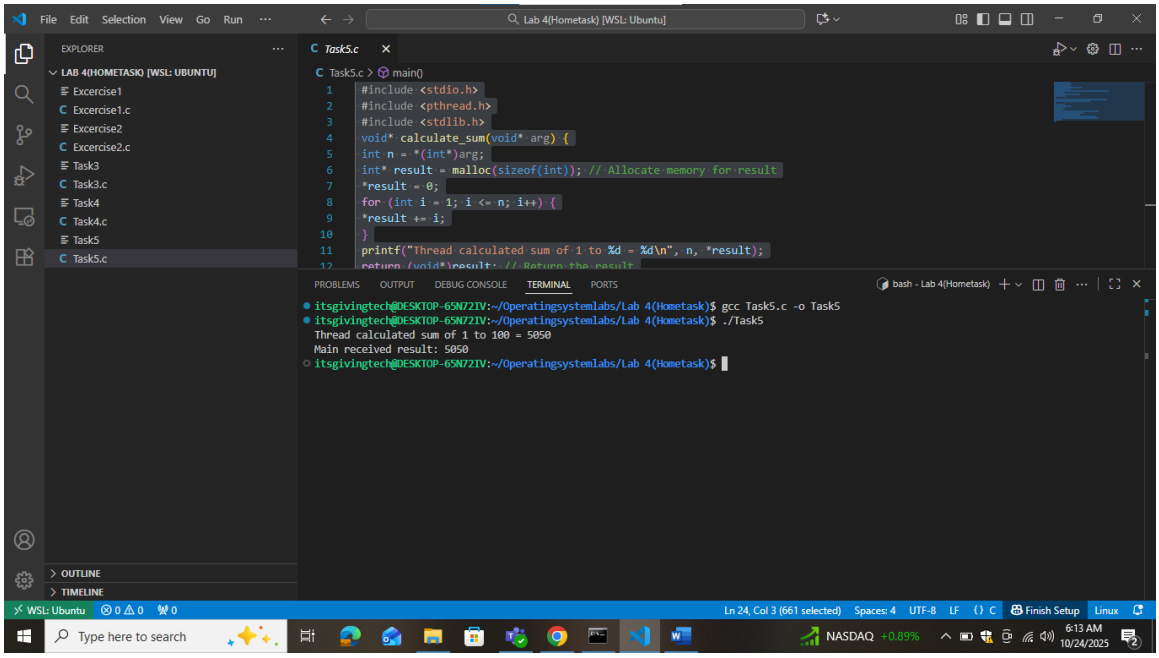
Program 5: Thread Return Values

Code:

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
void* calculate_sum(void* arg) {
int n = *(int*)arg;
int* result = malloc(sizeof(int));
*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;
}
int main() {
pthread_t thread_id;
int n = 100;
void* sum;
pthread_create(&thread_id, NULL, calculate_sum, &n);
pthread_join(thread_id, &sum);
printf("Main received result: %d\n", *(int*)sum);
```

```
free(sum);
return 0;
}
```

Output:



Exercise 1:

Write a program that:

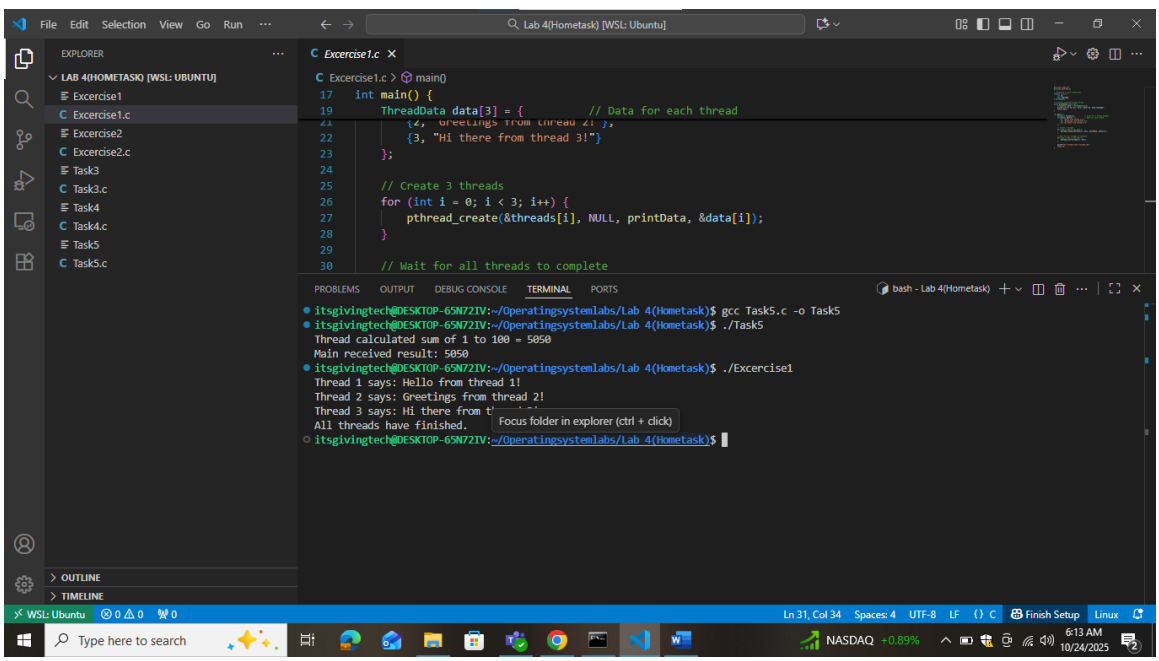
1. Creates 3 threads
2. Each thread prints its thread ID and a unique message
3. 3. Main thread waits for all threads to complete

Code:

```
#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;
}
```

```c
int main() {
    pthread_t threads[3];
    ThreadData data[3] = {
        {1, "Hello from thread 1!"},
        {2, "Greetings from thread 2!"},
        {3, "Hi there from thread 3!"}
    };


    for (int i = 0; i < 3; i++) {
        pthread_create(&threads[i], NULL, printData, &data[i]);
    }
    for (int i = 0; i < 3; i++) {
        pthread_join(threads[i], NULL);
    }

    printf("All threads have finished.\n");
    return 0;
}
```

Output:



Exercise 2:

Prime Number Checker:

1. Takes a number as input

2. Creates a thread that checks if the number is prime

3. Returns the result to the main thread

4. Main thread prints whether the number is prime or not

code:

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


typedef struct {
    int number;
    bool isPrime;
} ThreadData;


void* checkPrime(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    int n = data->number;

    if (n <= 1) {
        data->isPrime = false;
        return NULL;
    }

    data->isPrime = true;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            data->isPrime = false;
            break;
        }
    }

    return NULL;
}

int main() {
    pthread_t thread;
```

```c
    ThreadData data;


    printf("Enter a number: ");
    scanf("%d", &data.number);


    pthread_create(&thread, NULL, checkPrime, &data);

    pthread_join(thread, NULL);


    if (data.isPrime)
        printf("%d is a prime number.\n", data.number);
    else
        printf("%d is not a prime number.\n", data.number);

    return 0;
}
```

Output: