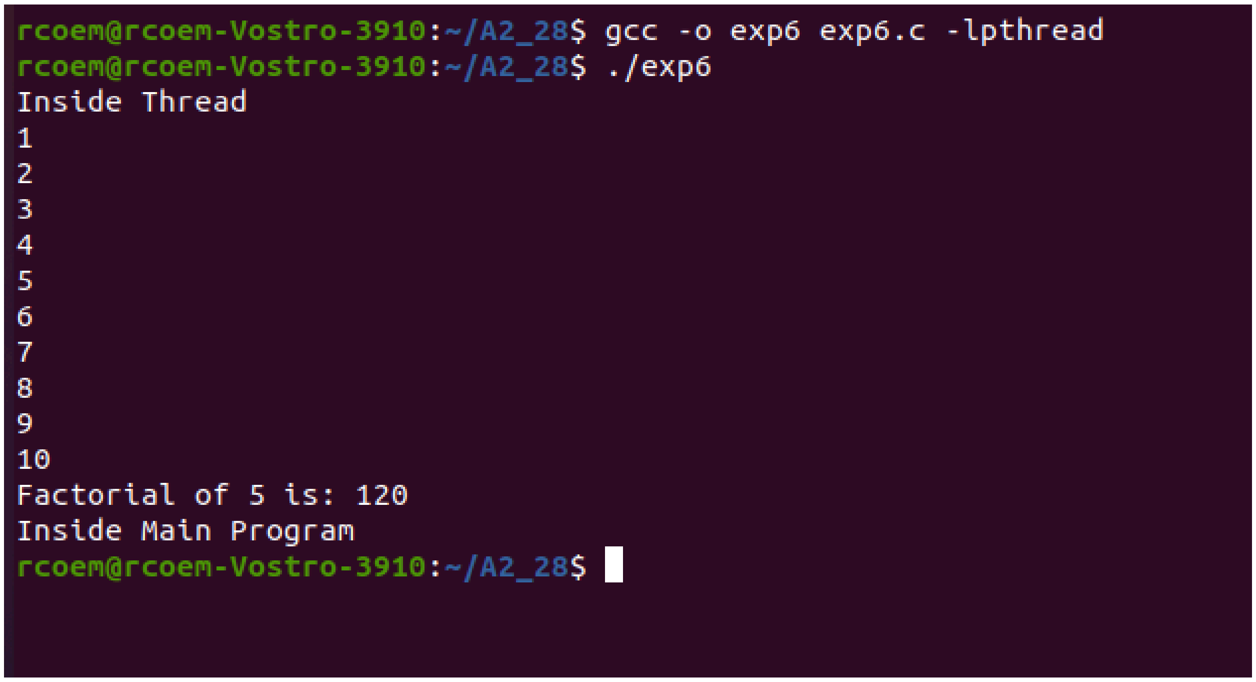
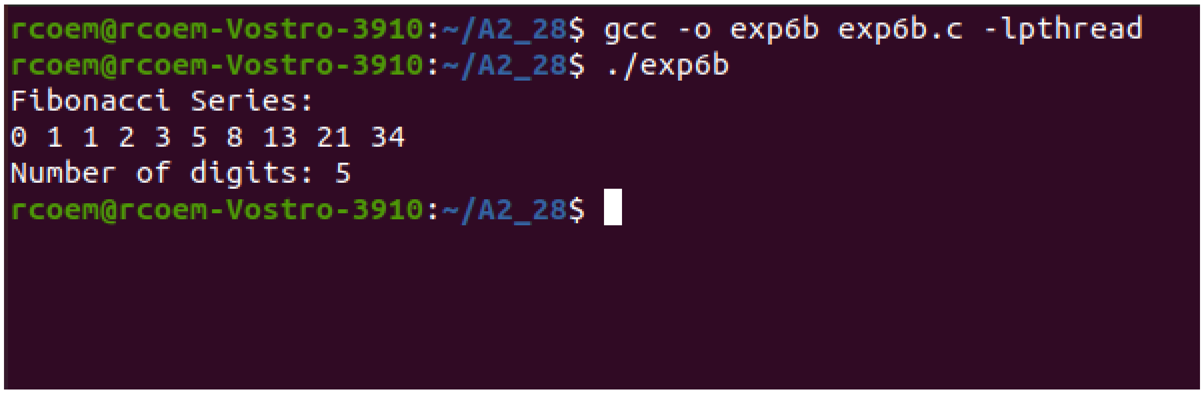
Name : Bhuvnesh Verma
Roll No : 28
Practical : 6
Aim : Multi - threading

Output for exp6 : create threads .



```
rcoem@rcoem-Vostro-3910:~/A2_28$ gcc -o exp6 exp6.c -lpthread
rcoem@rcoem-Vostro-3910:~/A2_28$ ./exp6
Inside Thread
1
2
3
4
5
6
7
8
9
10
Factorial of 5 is: 120
Inside Main Program
rcoem@rcoem-Vostro-3910:~/A2_28$
```

Output for exp6b : create threads and make two functions and call them.



```
rcoem@rcoem-Vostro-3910:~/A2_28$ gcc -o exp6b exp6b.c -lpthread
rcoem@rcoem-Vostro-3910:~/A2_28$ ./exp6b
Fibonacci Series:
0 1 1 2 3 5 8 13 21 34
Number of digits: 5
rcoem@rcoem-Vostro-3910:~/A2_28$
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

// Thread function to print numbers
void* thread_function(void *arg) {
    printf("Inside Thread \n");
    for (int i = 0; i < 10; i++) {
        printf("%d\n", i + 1);
        sleep(1); // Sleep for a second
    }
    return NULL;
}

// Thread function to calculate factorial
void* factorial(void *arg) {
    int n = *(int*)arg; // Get the number from the argument
    int result = 1;

    for (int i = 1; i <= n; i++) {
        result *= i;
    }

    printf("Factorial of %d is: %d\n", n, result);
    return NULL;
}

int main() {
    pthread_t a_thread; // Thread declaration

    // Create the number thread
    pthread_create(&a_thread, NULL, thread_function, NULL);

    // Wait for the number thread to finish
    pthread_join(a_thread, NULL);

    pthread_t fact_thread;
    int num = 5; // Number for which we want the factorial

    // Create the factorial thread
    pthread_create(&fact_thread, NULL, factorial, &num);

    // Wait for the factorial thread to finish
    pthread_join(fact_thread, NULL);

    printf("Inside Main Program\n");

    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

void* fibonacci(void *arg) {
    int n = *(int*)arg;
    int a = 0, b = 1, c;

    printf("Fibonacci Series:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", a);
        c = a + b;
        a = b;
        b = c;
        sleep(1);
    }
    printf("\n");
    return NULL;
}

void* count_digits(void *arg) {
    int number = *(int*)arg;
    int count = 0;

    while (number != 0) {
        number /= 10;
        count++;
    }

    printf("Number of digits: %d\n", count);
    return NULL;
}

int main() {
    pthread_t fib_thread;
    pthread_t count_thread;
    int n = 10;
    int num = 12345;
    pthread_create(&fib_thread, NULL, fibonacci, &n);pthread_join(fib_thread, NULL);
    pthread_create(&count_thread, NULL, count_digits, &num);pthread_join(count_thread,
NULL);

    return 0;
}
```