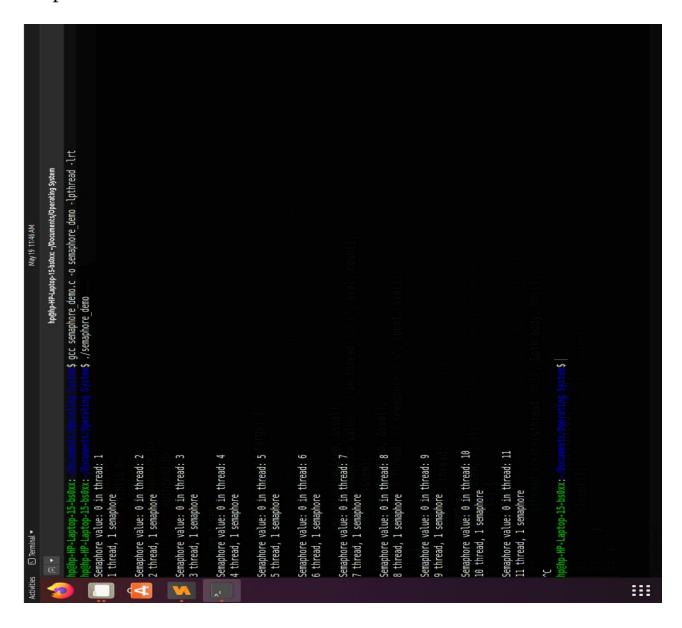
Assignment 9(a): Semaphores

Implementing semaphores in a C program for process synchronization:

```
To declare a semaphore:
    #include <semaphore.h>
    sem_t semaphore;
C function used to initialize a semaphore variable:
    #include <sempahore.h>
    int sem_init(sem_t *sem, int pshared, unsigned int value);
Here pshared is set to 0 for threads, 1 for processes.
C function for wait(mutex) function:
    int sem_wait(sem_t *sem);
C function for signal(mutex) function:
    int sem post(sem t *sem);
Sample code:
sem_t sem;
int count;
void* pth_body(void *args) {
    int sval;
    // using semaphore to allow only one thread to execute at one
time
    sem_wait(&sem); // decreases sem value by 1
    // critical section
    sem_getvalue(&sem, &sval);
    printf("Semaphore value: %d in thread: %d\n", sval, count);
    sem_post(&sem); // increases sem value by 1
    sem_getvalue(&sem, &sval);
```

```
printf("%d thread, %d semaphore\n\n", count, sval);
}
int main() {
    count = 0;
    sem_init(&sem, 0, 1); // initializing semaphore value
    /* rest of the main is same as pthread program*/
}
```

Output:



Assignment 9(b): Prime Check

Similar to Assignment 8