Course Title: **OPERATING SYSTEM**| Course Code: **CSE316**

Submitted by:

Sushant Kumar

Registration No: 12218023

Roll Number: 54

Section: K22ZU

Programme Name: B.Tech. CSE

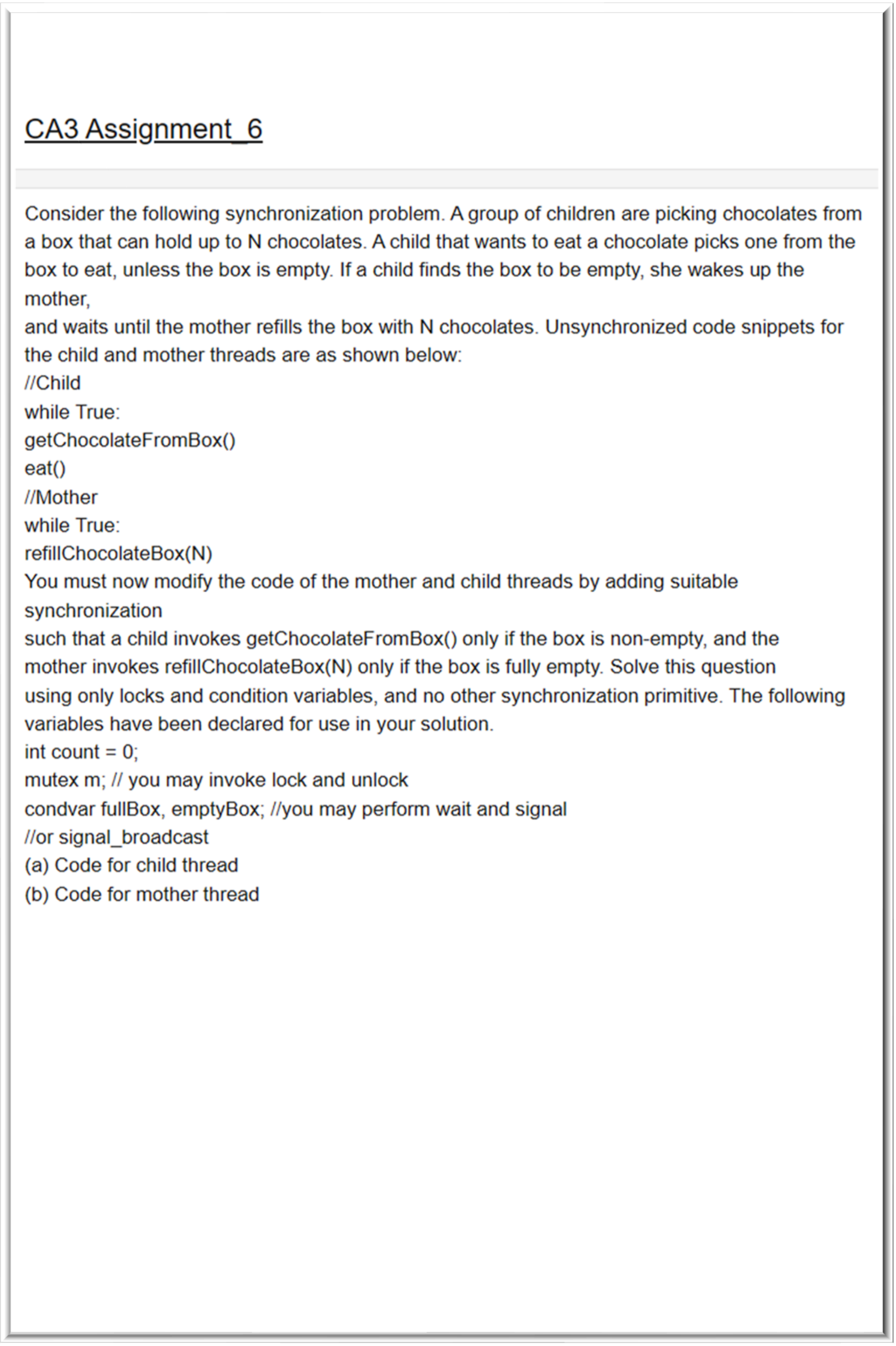
Under the Guidance of:

Anchal Kundal

School of Computer Science & Engineering

Lovely Professional University, Phagwara

Problem Assigned



CA3 Assignment_6

Consider the following synchronization problem. A group of children are picking chocolates from a box that can hold up to N chocolates. A child that wants to eat a chocolate picks one from the box to eat, unless the box is empty. If a child finds the box to be empty, she wakes up the mother,
and waits until the mother refills the box with N chocolates. Unsynchronized code snippets for the child and mother threads are as shown below:

```
//Child
while True:
getChocolateFromBox()
eat()
//Mother
while True:
refillChocolateBox(N)
```

You must now modify the code of the mother and child threads by adding suitable synchronization
such that a child invokes getChocolateFromBox() only if the box is non-empty, and the mother invokes refillChocolateBox(N) only if the box is fully empty. Solve this question using only locks and condition variables, and no other synchronization primitive. The following variables have been declared for use in your solution.

```
int count = 0;
mutex m; // you may invoke lock and unlock
condvar fullBox, emptyBox; //you may perform wait and signal
//or signal_broadcast
```

(a) Code for child thread
(b) Code for mother thread

Code Functionality

Process Synchronization: The program makes sure that different parts of the program work together smoothly. It prevents conflicts by allowing only one thing to happen at a time.

Producer-Consumer Problem: It's like a game where two characters, a mom, and a child, are involved. The mom keeps filling a box with chocolates, and the child keeps eating them.

Mutual Exclusion: Only one character (mom or child) is allowed to touch the chocolates at any moment. This prevents them from grabbing chocolates at the same time and making a mess.

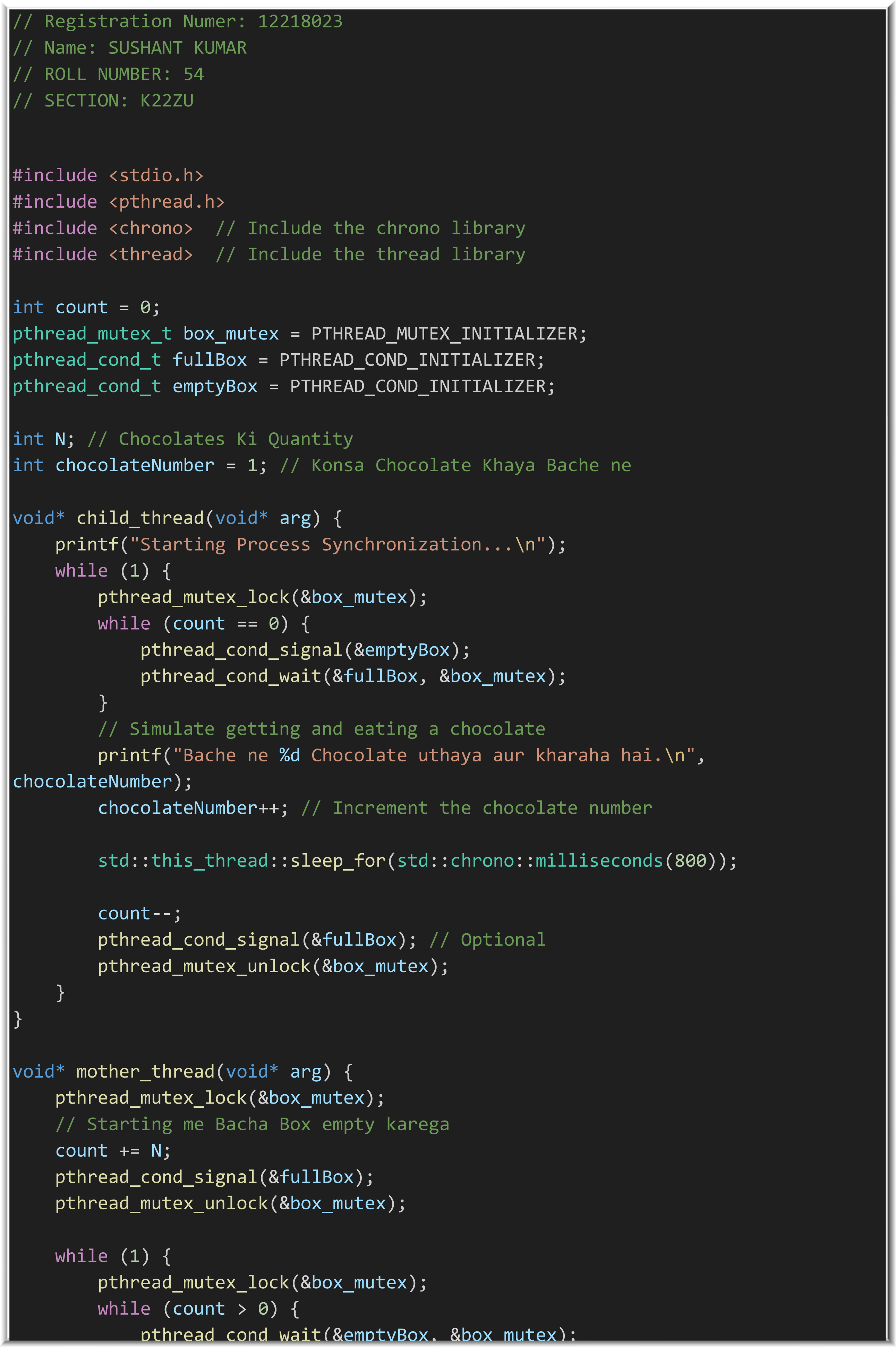
Condition Variables: The mom and child communicate with each other through signals. For example, when the box is full, the mom tells the child it's time to eat, and when the box is empty, the child tells the mom it's time to fill it again.

Thread Termination: The program waits for the mom and child to finish their job before ending the whole game.

Time Control: The program can control how much time the child waits before eating a chocolate and how long the mom takes to fill the box.

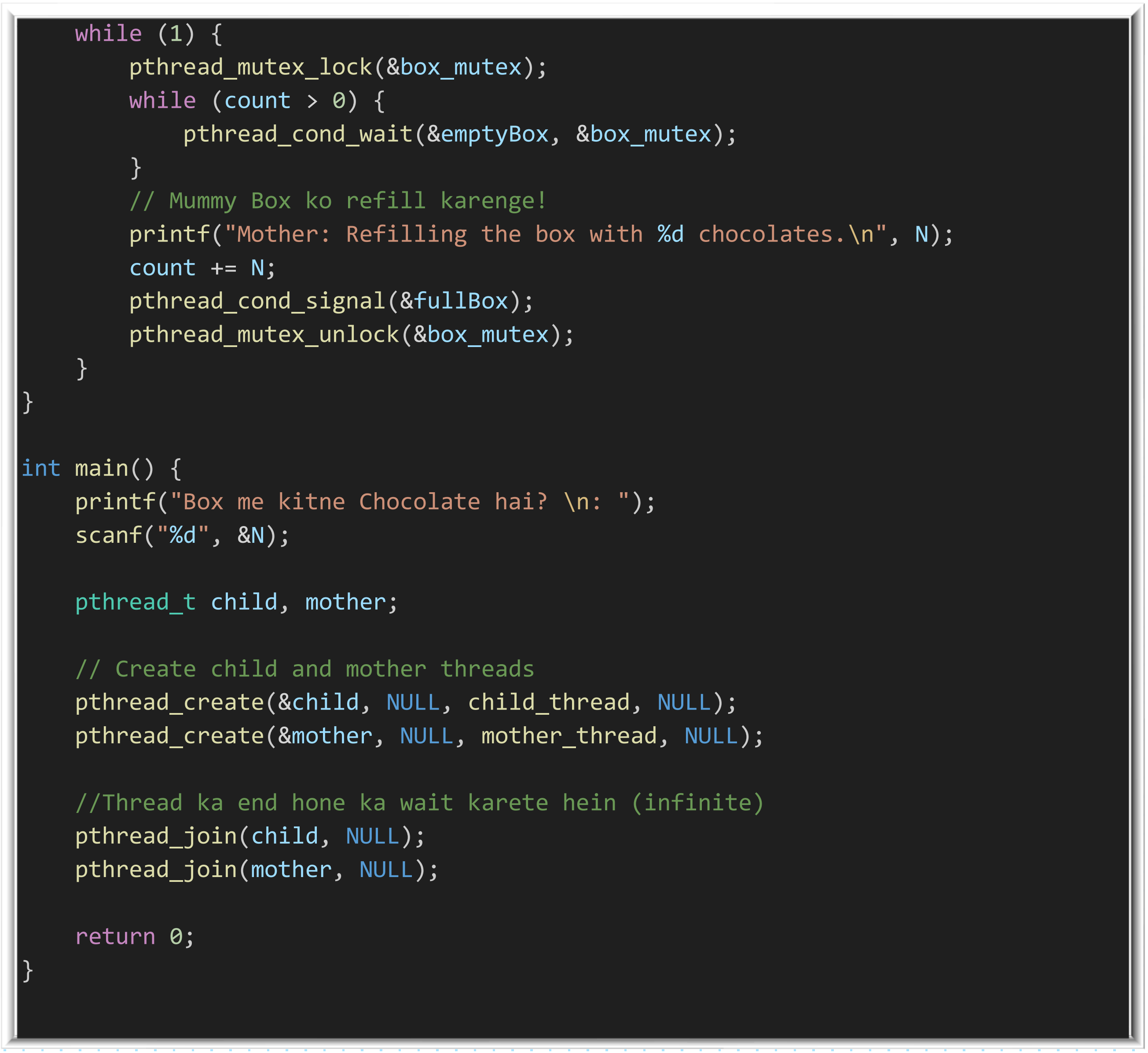
Basically, it's like a teamwork game between a mom and a child with rules to avoid fights, and they use signals to communicate when they need to do their tasks. The game waits for them to finish, and you can control how fast they work.

Code Snippet



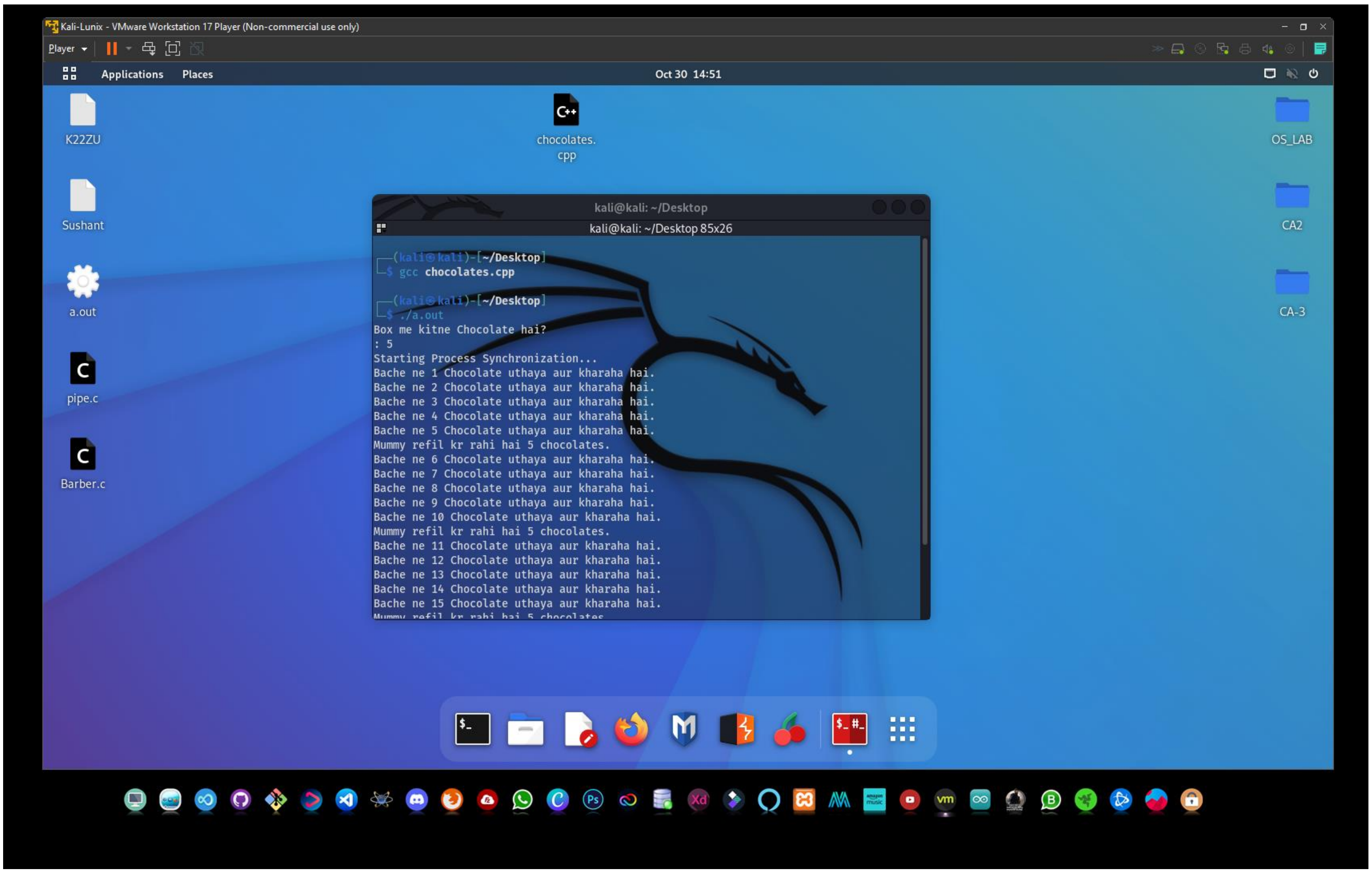
```
// Registration Numer: 12218023
// Name: SUSHANT KUMAR
// ROLL NUMBER: 54
// SECTION: K22ZU


#include <stdio.h>
#include <pthread.h>
#include <chrono>  // Include the chrono library
#include <thread>  // Include the thread library

int count = 0;
pthread_mutex_t box_mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t fullBox = PTHREAD_COND_INITIALIZER;
pthread_cond_t emptyBox = PTHREAD_COND_INITIALIZER;

int N; // Chocolates Ki Quantity
int chocolateNumber = 1; // Konsa Chocolate Khaya Bache ne

void* child_thread(void* arg) {
    printf("Starting Process Synchronization...\n");
    while (1) {
        pthread_mutex_lock(&box_mutex);
        while (count == 0) {
            pthread_cond_signal(&emptyBox);
            pthread_cond_wait(&fullBox, &box_mutex);
        }
        // Simulate getting and eating a chocolate
        printf("Bache ne %d Chocolate uthaya aur kharaha hai.\n",
chocolateNumber);
        chocolateNumber++; // Increment the chocolate number

        std::this_thread::sleep_for(std::chrono::milliseconds(800));

        count--;
        pthread_cond_signal(&fullBox); // Optional
        pthread_mutex_unlock(&box_mutex);
    }
}

void* mother_thread(void* arg) {
    pthread_mutex_lock(&box_mutex);
    // Starting me Bacha Box empty karega
    count += N;
    pthread_cond_signal(&fullBox);
    pthread_mutex_unlock(&box_mutex);

    while (1) {
        pthread_mutex_lock(&box_mutex);
        while (count > 0) {
            pthread_cond_wait(&emptyBox, &box_mutex);
```

Code Snippet



```c
    while (1) {
        pthread_mutex_lock(&box_mutex);
        while (count > 0) {
            pthread_cond_wait(&emptyBox, &box_mutex);
        }
        // Mummy Box ko refill karenge!
        printf("Mother: Refilling the box with %d chocolates.\n", N);
        count += N;
        pthread_cond_signal(&fullBox);
        pthread_mutex_unlock(&box_mutex);
    }
}

int main() {
    printf("Box me kitne Chocolate hai? \n: ");
    scanf("%d", &N);

    pthread_t child, mother;

    // Create child and mother threads
    pthread_create(&child, NULL, child_thread, NULL);
    pthread_create(&mother, NULL, mother_thread, NULL);

    //Thread ka end hone ka wait karete hein (infinite)
    pthread_join(child, NULL);
    pthread_join(mother, NULL);

    return 0;
}
```

Working Visual



```
┌──(kali㉿kali)-[~/Desktop]
└─$ gcc chocolates.cpp

┌──(kali㉿kali)-[~/Desktop]
└─$ ./a.out
Box me kitne Chocolate hai?
: 5
Starting Process Synchronization...
Bache ne 1 Chocolate uthaya aur kharaha hai.
Bache ne 2 Chocolate uthaya aur kharaha hai.
Bache ne 3 Chocolate uthaya aur kharaha hai.
Bache ne 4 Chocolate uthaya aur kharaha hai.
Bache ne 5 Chocolate uthaya aur kharaha hai.
Mummy refil kr rahi hai 5 chocolates.
Bache ne 6 Chocolate uthaya aur kharaha hai.
Bache ne 7 Chocolate uthaya aur kharaha hai.
Bache ne 8 Chocolate uthaya aur kharaha hai.
Bache ne 9 Chocolate uthaya aur kharaha hai.
Bache ne 10 Chocolate uthaya aur kharaha hai.
Mummy refil kr rahi hai 5 chocolates.
Bache ne 11 Chocolate uthaya aur kharaha hai.
Bache ne 12 Chocolate uthaya aur kharaha hai.
Bache ne 13 Chocolate uthaya aur kharaha hai.
Bache ne 14 Chocolate uthaya aur kharaha hai.
Bache ne 15 Chocolate uthaya aur kharaha hai.
Mummy refil kr rahi hai 5 chocolates
```

Thank You