# Documentation

## Main.java

public class Main {

    static class CountUpRunnable implements Runnable {

        @**Override**

        public void run() {

            for (int i = 1; i <= 20; i++) {

                System.out.println("Thread One: " + i);

                try {

                    Thread.sleep(500); *// Sleep for 500 milliseconds*

                } catch (InterruptedException e) {

                    e.printStackTrace();

                }

            }

        }

    }

    static class CountDownRunnable implements Runnable {

        @**Override**

        public void run() {

            for (int i = 20; i >= 0; i--) {

                System.out.println("Thread Two: " + i);

                try {

                    Thread.sleep(500); *// Sleep for 500 milliseconds*

                } catch (InterruptedException e) {

                    e.printStackTrace();

                }

            }

        }

    }

    public static void main(String[] args) {

        Thread countUp = new Thread(new CountUpRunnable());

        countUp.start();

        try {

            countUp.join();

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

        Thread countDown = new Thread(new CountDownRunnable());

        countDown.start();

        try {

            countDown.join();

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

}

## main.cpp

#include <iostream>

#include <thread>

#include <mutex>

std::mutex mutex;

void countUp()

{

for (int i = 1; i <= 20; ++i)

{

std::lock\_guard<std::mutex> lock(mutex);

std::cout << "Thread One: " << i << std::endl;

std::this\_thread::sleep\_for(std::chrono::milliseconds(500));

}

}

void countDown()

{

for (int i = 20; i >= 0; --i)

{

std::lock\_guard<std::mutex> lock(mutex);

std::cout << "Thread Two: " << i << std::endl;

std::this\_thread::sleep\_for(std::chrono::milliseconds(500));

}

}

int main()

{

std::thread t1(countUp);

std::thread t2(countDown);

t1.join();

t2.join();

return 0;

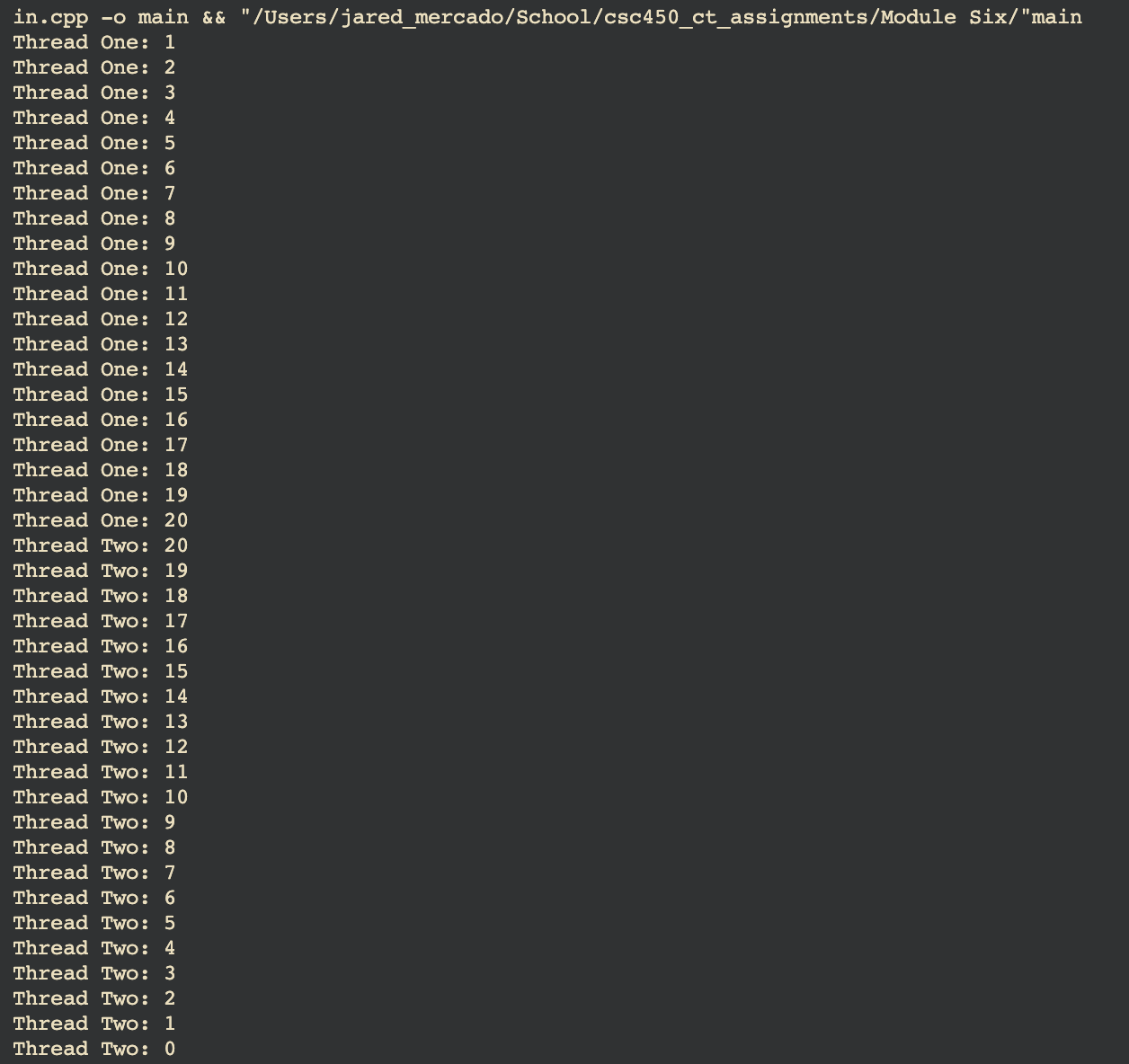
}

## Execution of Java Code

A black screen with white text

Description automatically generated

## Execution of C++ Code



## Java Pseudocode

Class Main {

Class CountUp implements Runnable {

void run() {

for(I till 20) {

print(I) every 0.5s;

}

}

}

Class CountDown implements Runnable {

void run() {

for(I from 20 till 0) {

print(I) every 0.5s;

}

}

}

public static void main(String[] args) {

Thread countUp = new Thread(new CountUp());

        countUp.start();

countUp.join();

Thread countDown = new Thread(new CountDown());

        countDown.start();

countDown.join();

}

}

## C++ Pseudocode

#include <iostream>

#include <thread>

#include <mutex>

std::mutex mutex;

void countUp{

for(i <= 20){

lock(mutex);

display(i);

sleep(0.5 seconds);

}

}

void countDown{

for(i >= 0){

lock(mutex);

display(i);

sleep(0.5 seconds);

}

}

int main(){

thread1(countUp);

thread2(countDown);

thread1 join and thread2 join;

return 0;

}

## Github

https://github.com/n0hb0dy/csc450\_ct\_assignments/tree/main/Module%20Eight

A screenshot of a computer

Description automatically generated