**Analysis**:

This application consists of 2 functions that are invoked form 2 threads in main, one function counts up to 20, and the other counts down once the first is executed. Due to the sequential nature of this application we will encounter performance issues with concurrency since we are consuming more of the CPU’s cores to execute code that could be executed from a single thread using a single core. However, this is a great way to learn about the challenges of creating multi threaded applications. In this application there are no vulnerabilities exhibited with use of strings because we only use hard coded strings in the applications output. We do not accept any inputs, so there is no need to sanitize the inputs of this application. The primitive data types we use are an int. Since we only set the value within the range 0 → 20 which will never surpass the limits of an int value.(hard to be specific since this range varies depending on compiler.) We also use thread. Threads pose the risk of issues due to concurrent code execution like race conditions or unpredictable output if some sequence of events are expected to occur. This can pose a security risk if a user is able to take advantage of this and exploit it to bypass a step in sequence. This code is a great example of this because increasing the max number can exhibit output that is out of sequence and strings will get mutilated in the process. This can be solved by making the functions atomic by using mutexes and semaphores to block the other thread from executing before the first thread is done with that task.

**Pseudo Code:**

final int MAX 20

void t\_count\_up()

{

for (int i = 0; i <= MAX; i++)

{

print("Thread: " + thread\_id() + " Value: " + i);

}

}

void t\_count\_down()

{

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

{

print("Thread: " + thread\_id() + " Value: " + i);

}

}

int main()

{

// Create two threads.

thread t1(t\_count\_up);

thread t2(t\_count\_down);

// Join the threads to this thread.

t1.join();

t2.join();

}

**Source Code:**

/\*\*

\* author: Alec McDaugale

\*

\* This program uses 2 threads to count up to 100 then counts back down to 0.

\*/

// Max number.

#define MAX 20

#include <iostream>

#include <thread>

using namespace std;

/\*\*

\* Function to count up from 0 to MAX.

\*/

void t\_count\_up()

{

for (int i = 0; i <= MAX; i++)

{

cout << "Thread: " << this\_thread::get\_id() << " Value: " << i << endl;

}

}

/\*\*

\* Function to count down from MAX to 0.

\*/

void t\_count\_down()

{

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

{

cout << "Thread: " << this\_thread::get\_id() << " Value: " << i << endl;

}

}

int main()

{

// Create two threads.

thread t1(t\_count\_up);

thread t2(t\_count\_down);

// Join the threads to this thread.

t1.join();

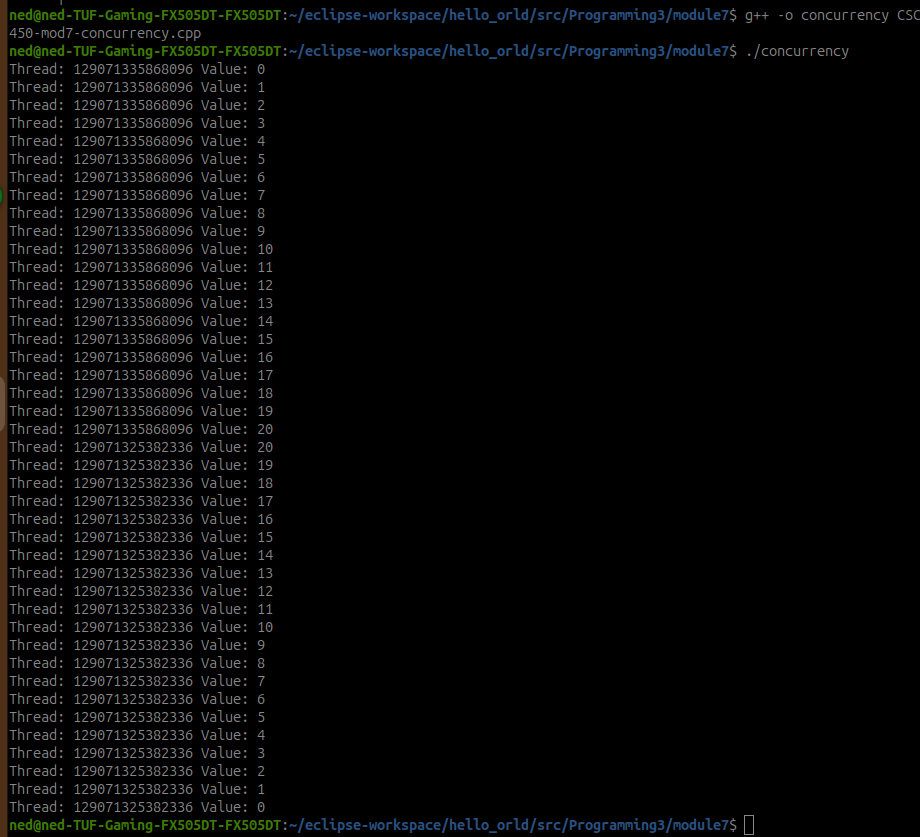
t2.join();

// Return success.

return 0;

}

**ScreenShot:**



**Github:**

<https://github.com/amcdaugale/Programming3/tree/main/module7>

