Assignment 2: Threading   
  
Contributors:   
  
JEBREEL EMAD ABEDAL KAREEM MAJDALWIEH

ISLAM ZAID SULEIMAN ALSHQEIRAT

SHATHA HOSSAM ISSA MUSTAFA

Task1:

Screenshot of the main code :

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Task2:

Screenshot of the thread function(s):

Text

Description automatically generated

Task3:

Screenshot highlighting the parts of the code that were added to make the code thread-safe, with explanations on the need for them:  
  
  
Text

Description automatically generated

Declaring 2 locks

Global   
variables

Text

Description automatically generated

UnLock “lock”

Lock “lock”

Critical section 2

UnLock “Flock”

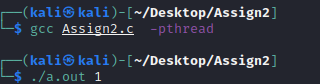
Lock “Flock”

Critical section 1

1. Flock: we used mutex Flock to lock the critical section 1, which might happen on the global pointer \*wfptr since the threads are writing on the file  
   \*note: this might be unnecessary depending on the low level definition of fprintf() but we added this lock just in case.
2. Lock: we used this lock on critical section 2, to save the global variables Even, Odd, Total from racing between threads, because these variables are shared between all the threads and all the threads (more than 1) write on them.  
     
   - all other global variables are thread-safe as we noted in the screenshot above (comments).

Task4:

Screenshot of the output of the two versions of your code (thread-safe vs. non-thread-safe), when running passing the following number of threads (T): 1, 2, 4, 8, 16, 32.   
  
thread safe :

T=1:  
  
 A screenshot of a computer screen

Description automatically generated with medium confidence

T=2:

A screenshot of a computer

Description automatically generated

T=4:  
  
A screenshot of a computer

Description automatically generated with medium confidence

T=8:  
Graphical user interface, text

Description automatically generated

T=16:

A screenshot of a computer

Description automatically generated with medium confidence

T=32:  
Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. **Non-thread safe :**

T=1:  
A screenshot of a computer screen

Description automatically generated with medium confidence  
  
T=2:  
A screenshot of a computer

Description automatically generated with medium confidence  
  
T=4:  
A screenshot of a computer

Description automatically generated with medium confidence  
  
T=8:  
A screenshot of a computer

Description automatically generated with medium confidence

T=16:  
A screenshot of a computer screen

Description automatically generated with medium confidence  
  
T=32:  
A screenshot of a computer

Description automatically generated with medium confidenceGraphical user interface, text, application

Description automatically generated  
  
Task5:  
we made 3 tests to make sure our conclusion is accurate:  
  
Test1:  
  
 Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated  
Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated  
  
  
  
  
we made a chart to make the result more clear:  
Chart

Description automatically generated

-to be continued ….

**Test 2:**Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generated **Chart2:**Chart, line chart

Description automatically generated **Test3:**Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generated **chart3:**Chart, line chart

Description automatically generated

**-Our conclusion :  
our machine should have near 4 cores with 1 hyper-thread each   
or something near these numbers,  
because when our program works on parallelism, each cpu core should work on 1 software thread all at the same time, which means that whan you double up the number of software threads, the execution time should cut in half (roughly).  
in our tests we noticed that whenever we use more than 4 threads we don’t see any major improvement on the execution time, and sometimes the time increases instead of improving(the law of diminishing returns).  
  
  
  
  
  
- machine actual properties:**Text

Description automatically generated