Assignment 2: Threading

Contributors:

JEBREEL EMAD ABEDAL KAREEM MAJDALWIEH 133050

ISLAM ZAID SULEIMAN ALSHQEIRAT 133034

SHATHA HOSSAM ISSA MUSTAFA 133844

Task1:

Screenshot of the main code :

```c
int main(int argc, char* argv[])
{

    //*************************************************************************************************************************

    wfptr = fopen("out.txt", "w");
    rfptr = fopen("in.txt", "r");


    fscanf(rfptr, "%d", &n); //reading n from file

    t = atoi(argv[1]);    //reading t from Argv (passed in the command line)



//*****************************************************************************************************************************

    matrix1 = (int**)malloc(n * sizeof(int*));             //allocating double pointer to make a 2D Array(matrix)  matrix1[n][n]
    for (int i = 0; i < n; i++)
        matrix1[i] = (int*)malloc(n * sizeof(int));


    matrix2 = (int**)malloc(n * sizeof(int*));            //allocating double pointer to make a 2D Array(matrix)  matrix2[n][n]
    for (int i = 0; i < n; i++)
        matrix2[i] = (int*)malloc(n * sizeof(int));

    matrix3 = (int**)malloc(n * sizeof(int*));             //allocating double pointer to make a 2D Array(matrix)  matrix3[n][n]
    for (int i = 0; i < n; i++)
        matrix3[i] = (int*)malloc(n * sizeof(int));
```

```c
  //************************************************************************************************************************...


  for (int i = 0; i < n; i++)                          //reading matrix1 from the input file
      for (int j = 0; j < n; j++)
          fscanf(rfptr, "%d", &matrix1[i][j]);



  for (int i = 0; i < n; i++)                           //reading matrix2 from the input file
      for (int j = 0; j < n; j++)
          fscanf(rfptr, "%d", &matrix2[i][j]);
```

```c
pthread_t* threads = malloc(t * sizeof(pthread_t));    //allocating a dynamic array of type pthread


int ef; // for exit

for (intptr_t i = 0; i < t; i++)        //a loop that executes t times
{

    ef = pthread_create(&threads[i], NULL, multi, (void*)i); //creating threads which execute the function "multi" and passing thread number (id) to the function



    if (ef)  //Errors Handling
    {
        printf("\n\n A problem has occured \n\n");
        exit(-1);
    }
}


for (int i = 0; i < t; i++)  //JOIN
{
    pthread_join(threads[i], NULL);
}
```

```c
    printf("numOfEven= %d, numOfOdd=%d, totalCells=%d ", Even, Odd, Total);
    fprintf(wfptr, "numOfEven= %d, numOfOdd=%d, totalCells=%d \n\n", Even, Odd, Total);



    //*************************************************************************************************************

                                                    //deallocating the dynamic arrays
    for (int i = 0; i < n; i++)
    {
        free(matrix1[i]);
        free(matrix2[i]);
        free(matrix3[i]);
    }

    free(matrix1);
    free(matrix2);
    free(matrix3);
    free(threads);


    fclose(rfptr);
    fclose(wfptr);
    //fclose(tfptr);

    pthread_exit(NULL);  //to exit the main thread
}
```

Task2:

Screenshot of the thread function(s):

```
void* multi(void* x)
{

    //important : we distrebuted the work on the threads equally by rows, n/t finds how many rows each thread works on

    int b = (int64_t)x;   //casting from void* to int, x resembles the thread number(ID)
    int s = b * (n / t);       //start loop , calculated from the thread number
    int e = s + (n / t);       //end loop   ,calculated from the thread number


    printf("ThreadID=%d, startLoop=%d, endLoop=%d\n\n", b, s, e);

    fprintf(wfptr, "ThreadID=%d, startLoop=%d, endLoop=%d\n\n", b, s, e);


    int sum = 0;
    for (int k = s; k < e; k++)
        for (int i = 0; i < n; i++)
        {
            sum = 0;
            for (int j = 0; j < n; j++)
                sum += matrix1[k][j] * matrix2[j][i];


            matrix3[k][i] = sum;



            if (sum % 2 == 0)
                Even++;
            else
                Odd++;
            Total++;



        }



}
```

Task3:

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

```c
int Even = 0, Odd = 0, Total = 0;   //(write)
int t;                       //number of threads (read only)
int n;                       //Matrix Size       (read only)
int** matrix1;               //input matrix  (read only)
int** matrix2;               //input matrix  (read only)
int** matrix3;               //output matrix (write)
FILE* wfptr;                 //(write)
FILE* rfptr;
```

Global variables

```c
pthread_mutex_t lock;        //Lock for Even,Odd,Total
pthread_mutex_t Flock;       //Lock for Wfptr
```

Declaring 2 locks

```c
void* multi(void* x)
{

    //important : we distrebuted the work on the threads equally by rows, n/t finds how many rows each thread works on

    int b = (int64_t)x;   //casting from void* to int, x resembles the thread number(ID)
    int s = b * (n / t);       //start loop , calculated from the thread number
    int e = s + (n / t);       //end loop   ,calculated from the thread number


    printf("ThreadID=%d, startLoop=%d, endLoop=%d\n\n", b, s, e);
    pthread_mutex_lock(&Flock);
    fprintf(wfptr, "ThreadID=%d, startLoop=%d, endLoop=%d\n\n", b, s, e);
    pthread_mutex_unlock(&Flock);

    int sum = 0;
    for (int k = s; k < e; k++)
        for (int i = 0; i < n; i++)
        {
            sum = 0;
            for (int j = 0; j < n; j++)
                sum += matrix1[k][j] * matrix2[j][i];

            matrix3[k][i] = sum;

            pthread_mutex_lock(&lock);
            if (sum % 2 == 0)
                Even++;
            else
                Odd++;
            Total++;
            pthread_mutex_unlock(&lock);
        }


}
```

Lock “Flock”

Critical section 1

UnLock “Flock”

UnLock “lock”

Critical section 2

UnLock “lock”

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.

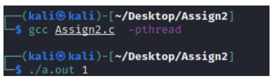
   - all other global variables are thread-safe as we noted in the screenshot above (comments).

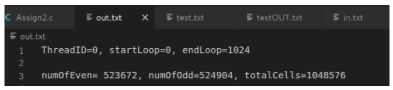
Task4:

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 :

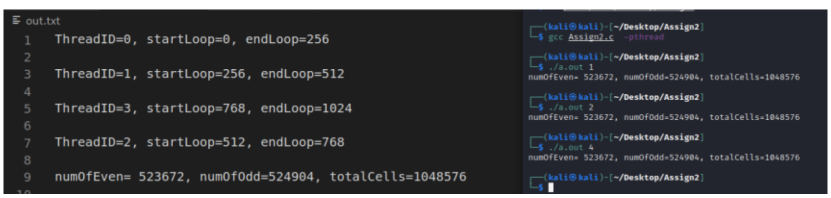
T=1:



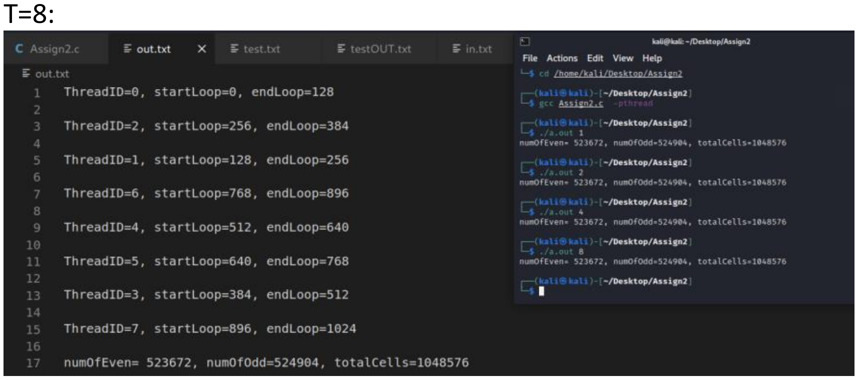


T=2:

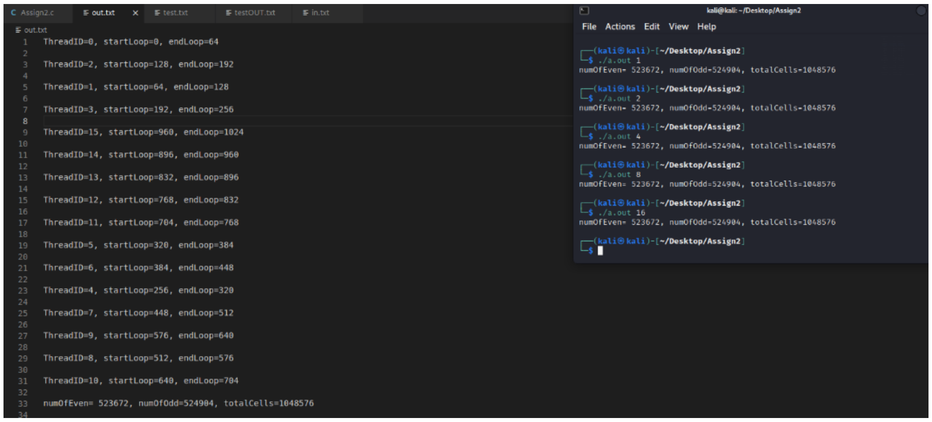
T=4:



T=8:

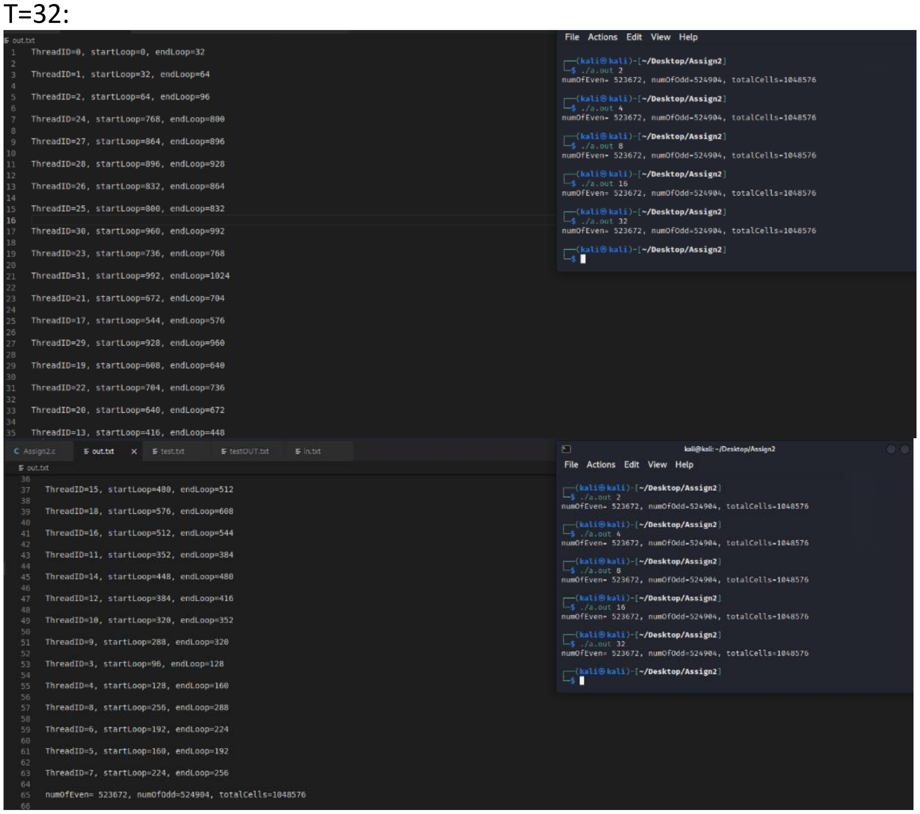


T=16:



ThreadID=0, startLoop=0, endLoop=64
ThreadID=2, startLoop=128, endLoop=192
ThreadID=1, startLoop=64, endLoop=128
ThreadID=3, startLoop=192, endLoop=256
ThreadID=15, startLoop=960, endLoop=1024
ThreadID=14, startLoop=896, endLoop=960
ThreadID=13, startLoop=832, endLoop=896
ThreadID=12, startLoop=768, endLoop=832
ThreadID=11, startLoop=704, endLoop=768
ThreadID=5, startLoop=320, endLoop=384
ThreadID=6, startLoop=384, endLoop=448
ThreadID=4, startLoop=256, endLoop=320
ThreadID=7, startLoop=448, endLoop=512
ThreadID=9, startLoop=576, endLoop=640
ThreadID=8, startLoop=512, endLoop=576
ThreadID=10, startLoop=640, endLoop=704
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
File Actions Edit View Help
./a.out 1
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
./a.out 2
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
./a.out 4
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
./a.out 8
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
./a.out 16
numOfEven= 523672, numOfOdd=524904, totalCells=1048576

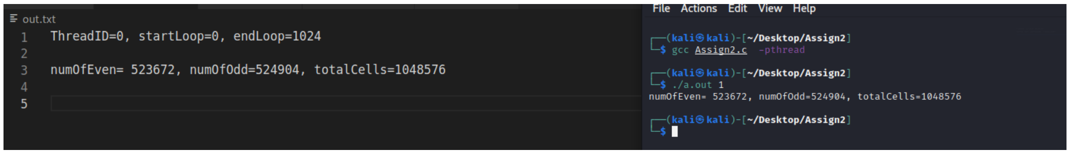
T=32:



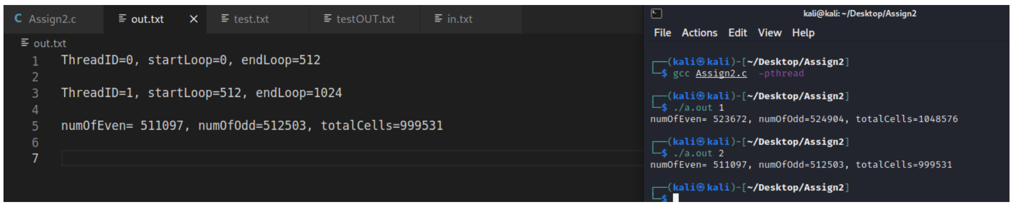
ThreadID=0, startLoop=0, endLoop=32
ThreadID=1, startLoop=32, endLoop=64
ThreadID=2, startLoop=64, endLoop=96
ThreadID=24, startLoop=768, endLoop=800
ThreadID=27, startLoop=864, endLoop=896
ThreadID=28, startLoop=896, endLoop=928
ThreadID=26, startLoop=832, endLoop=864
ThreadID=25, startLoop=800, endLoop=832
ThreadID=30, startLoop=960, endLoop=992
ThreadID=23, startLoop=736, endLoop=768
ThreadID=31, startLoop=992, endLoop=1024
ThreadID=21, startLoop=672, endLoop=704
ThreadID=17, startLoop=544, endLoop=576
ThreadID=29, startLoop=928, endLoop=960
ThreadID=19, startLoop=608, endLoop=640
ThreadID=22, startLoop=704, endLoop=736
ThreadID=20, startLoop=640, endLoop=672
ThreadID=13, startLoop=416, endLoop=448
ThreadID=15, startLoop=480, endLoop=512
ThreadID=18, startLoop=576, endLoop=608
ThreadID=16, startLoop=512, endLoop=544
ThreadID=11, startLoop=352, endLoop=384
ThreadID=14, startLoop=448, endLoop=480
ThreadID=12, startLoop=384, endLoop=416
ThreadID=10, startLoop=320, endLoop=352
ThreadID=9, startLoop=288, endLoop=320
ThreadID=3, startLoop=96, endLoop=128
ThreadID=4, startLoop=128, endLoop=160
ThreadID=8, startLoop=256, endLoop=288
ThreadID=6, startLoop=192, endLoop=224
ThreadID=5, startLoop=160, endLoop=192
ThreadID=7, startLoop=224, endLoop=256
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
File Actions Edit View Help
./a.out 2
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
./a.out 4
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
./a.out 8
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
./a.out 16
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
./a.out 32
numOfEven= 523672, numOfOdd=524904, totalCells=1048576

2- Non-thread safe :

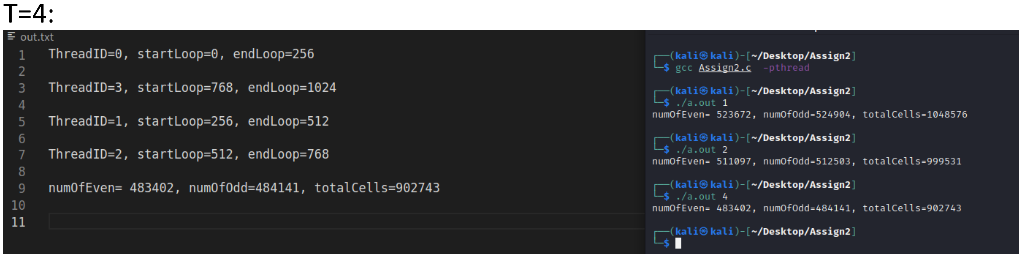
T=1:



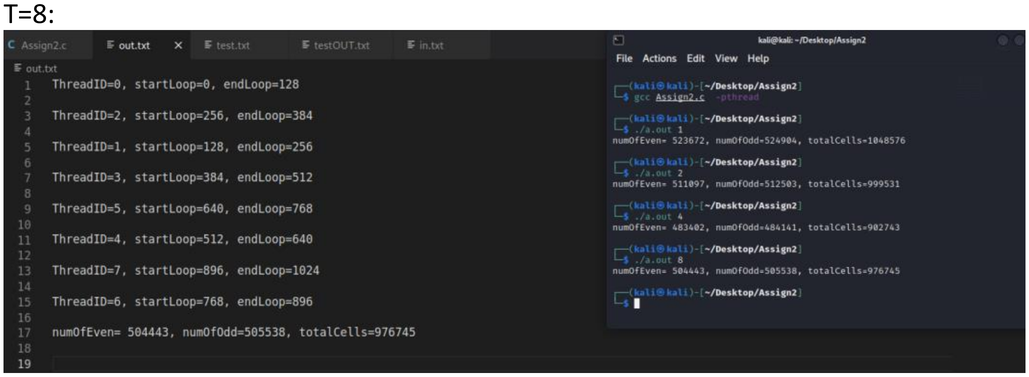
T=2:



T=4:



T=8:



T=16:

```
out.txt
ThreadID=0, startLoop=0, endLoop=64
ThreadID=2, startLoop=128, endLoop=192
ThreadID=1, startLoop=64, endLoop=128
ThreadID=3, startLoop=192, endLoop=256
ThreadID=15, startLoop=960, endLoop=1024
ThreadID=5, startLoop=320, endLoop=384
ThreadID=4, startLoop=256, endLoop=320
ThreadID=12, startLoop=768, endLoop=832
ThreadID=6, startLoop=384, endLoop=448
ThreadID=11, startLoop=704, endLoop=768
ThreadID=8, startLoop=512, endLoop=576
ThreadID=14, startLoop=896, endLoop=960
ThreadID=13, startLoop=832, endLoop=896
ThreadID=7, startLoop=448, endLoop=512
ThreadID=10, startLoop=640, endLoop=704
ThreadID=9, startLoop=576, endLoop=640
numOfEven= 511899, numOfOdd=513112, totalCells=982755
```

```
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 1
numOfEven= 523672, numOfOdd=524904, totalCells=1048576

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 2
numOfEven= 511097, numOfOdd=512503, totalCells=999531

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 4
numOfEven= 483402, numOfOdd=484141, totalCells=902743

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 8
numOfEven= 504443, numOfOdd=505538, totalCells=976745

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 16
numOfEven= 511899, numOfOdd=513112, totalCells=982755

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ 
```

T=32:

```
Assign2.c   out.txt ×   test.txt   testOUT.txt   in.txt
out.txt
ThreadID=0, startLoop=0, endLoop=32
ThreadID=4, startLoop=128, endLoop=160
ThreadID=1, startLoop=32, endLoop=64
ThreadID=2, startLoop=64, endLoop=96
ThreadID=24, startLoop=768, endLoop=800
ThreadID=25, startLoop=800, endLoop=832
ThreadID=28, startLoop=896, endLoop=928
ThreadID=26, startLoop=832, endLoop=864
ThreadID=23, startLoop=736, endLoop=768
ThreadID=29, startLoop=928, endLoop=960
ThreadID=30, startLoop=960, endLoop=992
ThreadID=19, startLoop=608, endLoop=640
ThreadID=20, startLoop=640, endLoop=672
ThreadID=21, startLoop=672, endLoop=704
ThreadID=15, startLoop=480, endLoop=512
ThreadID=17, startLoop=544, endLoop=576
ThreadID=16, startLoop=512, endLoop=544
ThreadID=11, startLoop=352, endLoop=384
```

```
kali@kali: ~/Desktop/Assign2
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 2
numOfEven= 511097, numOfOdd=512503, totalCells=999531

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 4
numOfEven= 483402, numOfOdd=484141, totalCells=902743

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 8
numOfEven= 504443, numOfOdd=505538, totalCells=976745

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 16
numOfEven= 511899, numOfOdd=513112, totalCells=982755

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 32
numOfEven= 507214, numOfOdd=508328, totalCells=987139

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ 
```

```
Assign2.c   out.txt ×   test.txt   testOUT.txt   in.txt
out.txt
ThreadID=13, startLoop=416, endLoop=448
ThreadID=27, startLoop=864, endLoop=896
ThreadID=12, startLoop=384, endLoop=416
ThreadID=10, startLoop=320, endLoop=352
ThreadID=31, startLoop=992, endLoop=1024
ThreadID=22, startLoop=704, endLoop=736
ThreadID=18, startLoop=576, endLoop=608
ThreadID=14, startLoop=448, endLoop=480
ThreadID=5, startLoop=160, endLoop=192
ThreadID=7, startLoop=224, endLoop=256
ThreadID=9, startLoop=288, endLoop=320
ThreadID=6, startLoop=192, endLoop=224
ThreadID=3, startLoop=96, endLoop=128
ThreadID=8, startLoop=256, endLoop=288
numOfEven= 507214, numOfOdd=508328, totalCells=987139
```


```
kali@kali: ~/Desktop/Assign2
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 2
numOfEven= 511097, numOfOdd=512503, totalCells=999531

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 4
numOfEven= 483402, numOfOdd=484141, totalCells=902743

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 8
numOfEven= 504443, numOfOdd=505538, totalCells=976745

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 16
numOfEven= 511899, numOfOdd=513112, totalCells=982755

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ ./a.out 32
numOfEven= 507214, numOfOdd=508328, totalCells=987139

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ 
```

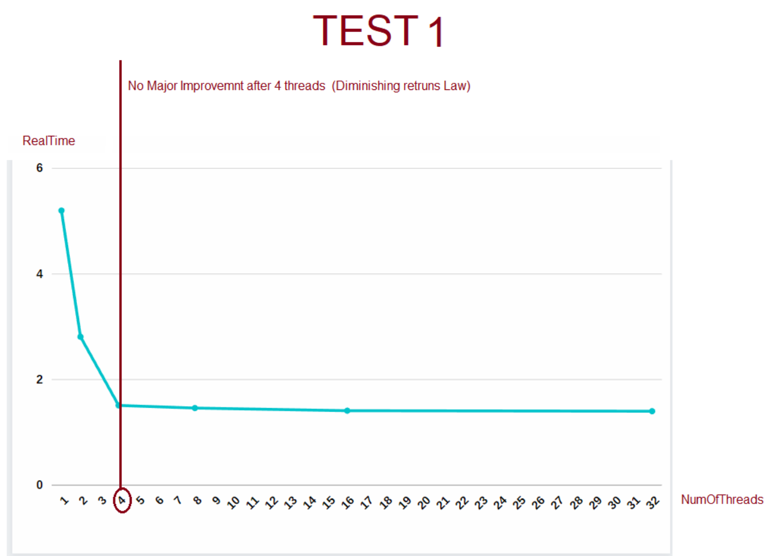
Task5:

we made 3 tests to make sure our conclusion is accurate:

Test1:

```
┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ time ./a.out 1
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     5.20s
user     5.19s
sys      0.01s
cpu      99%
└─$ time ./a.out 2
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     2.81s
user     5.22s
sys      0.05s
cpu      187%
└─$ time ./a.out 4
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     1.51s
user     5.02s
sys      0.08s
cpu      338%
└─$ time ./a.out 8
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     1.46s
user     4.79s
sys      0.06s
cpu      331%
└─$ time ./a.out 16
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     1.41s
user     4.73s
sys      0.06s
cpu      339%
└─$ time ./a.out 32
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     1.40s
user     4.64s
sys      0.08s
cpu      337%
```

we made a chart to make the result more clear:



-to be continued ....

Test 2:

```
└─$ time ./a.out 1
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real    5.34s
user    5.31s
sys     0.01s
cpu     99%
```

```
└─$ time ./a.out 2
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real    3.06s
user    5.76s
sys     0.04s
cpu     189%
```

```
└─$ time ./a.out 4
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real    1.88s
user    5.33s
sys     0.14s
cpu     290%
```

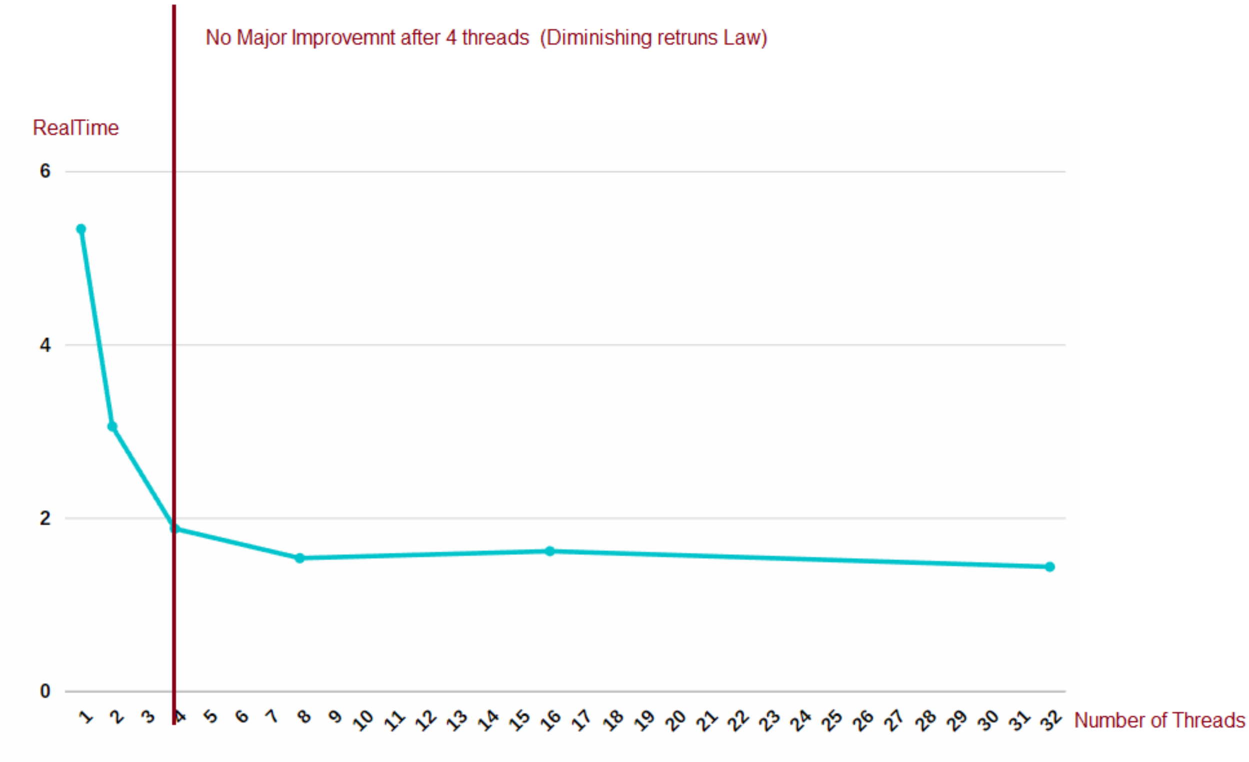
```
└─$ time ./a.out 8
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real    1.54s
user    4.89s
sys     0.11s
cpu     324%
```

```
└─$ time ./a.out 16
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real    1.62s
user    5.04s
sys     0.11s
cpu     319%
```

```
└─$ time ./a.out 32
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real    1.44s
user    4.82s
sys     0.10s
cpu     341%
```

Chart2:

TEST 2


No Major Improvemnt after 4 threads (Diminishing retruns Law)
RealTime
6
4
2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Number of Threads

Test3:

```
┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ time ./a.out 1
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     6.14s
user     6.10s
sys      0.02s
cpu      99%
```

```
└─$ time ./a.out 2
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     3.18s
user     6.01s
sys      0.06s
cpu      190%
```

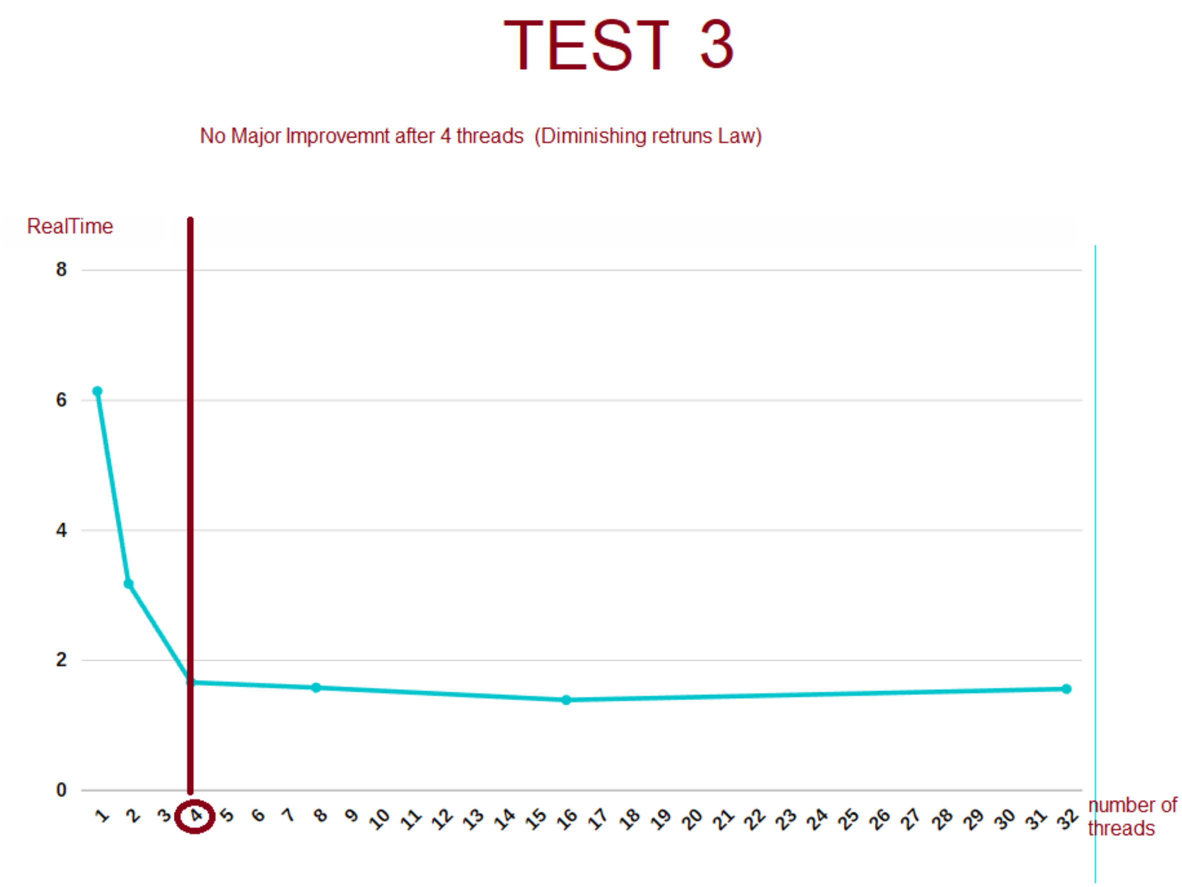
```
└─$ time ./a.out 4
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     1.66s
user     5.06s
sys      0.14s
cpu      313%
```

```
└─$ time ./a.out 8
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     1.58s
user     5.00s
sys      0.09s
cpu      321%
```

```
└─$ time ./a.out 16
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     1.39s
user     4.72s
sys      0.06s
cpu      342%
```

```
└─$ time ./a.out 32
numOfEven= 523672, numOfOdd=524904, totalCells=1048576
real     1.56s
user     4.96s
sys      0.07s
cpu      323%
```

chart3:


TEST 3
No Major Improvemnt after 4 threads (Diminishing retruns Law)
RealTime
8
6
4
2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
number of threads

-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:

```
┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ lscpu
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  4
  On-line CPU(s) list:   0-3
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz
    CPU family:          6
    Model:               158
    Thread(s) per core:  1
    Core(s) per socket:  4
    Socket(s):           1
    Stepping:            10
    BogoMIPS:            5615.99
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
                          mca cmov pat pse36 clflush mmx fxsr sse sse2 ht sys
                         call nx rdtscp lm constant_tsc rep_good nopl xtopolo
                         gy nonstop_tsc cpuid tsc_known_freq pni pclmulqdq ss
                         se3 cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes
                         xsave avx rdrand hypervisor lahf_lm abm 3dnowprefetc
                         h invpcid_single pti fsgsbase avx2 invpcid rdseed cl
                         flushopt md_clear flush_l1d
Virtualization features:
  Hypervisor vendor:     KVM
  Virtualization type:   full
Caches (sum of all):
  L1d:                   128 KiB (4 instances)
  L1i:                   128 KiB (4 instances)
  L2:                    1 MiB (4 instances)
  L3:                    36 MiB (4 instances)
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0-3
Vulnerabilities:
  Itlb multihit:         KVM: Mitigation: VMX unsupported
  L1tf:                  Mitigation; PTE Inversion
  Mds:                   Mitigation; Clear CPU buffers; SMT Host state unknow
                         n
  Meltdown:              Mitigation; PTI
  Spec store bypass:     Vulnerable
  Spectre v1:            Mitigation; usercopy/swapgs barriers and __user poin
                         ter sanitization
  Spectre v2:            Mitigation; Full generic retpoline, STIBP disabled,
                         RSB filling
  Srbds:                 Unknown: Dependent on hypervisor status
  Tsx async abort:       Not affected

┌──(kali㉿kali)-[~/Desktop/Assign2]
└─$ █
```