**Goal:** Getting experience in decomposing the solution in such way that multiple threads improve performance, i.e. decrease time to perform a given task by taking advantage of multiple processors in the system.

**Introduction:** In this project, you will design programming solutions for matrix multiplication that take advantage of multiple threads executing in parallel.

Let matrix $A$ be $n \times m$ integer matrix, and let $B$ be $m \times p$ integer matrix. Then $A \times B$ results in matrix $C \times p$. Code below provides the algorithm for multiplication $A \times B = C$.

```c
for (i=0; i < n; i++)
    for (j=0; j < p; j++)
    {
        c[i][j]=0;
        for (k=0; k < m; k++)
            c[i][j] += a[i][k]*b[k][j];
    }
```

You should develop algorithm for matrix multiplication that provides possibility that 2 or more threads can be executed (i.e. be running) in parallel. Try to develop algorithm that works for $n$ threads, where $n = 2, 3, 4, \ldots$

**Assignment:** In your program matrix $A$ should be $6000 \times 3000$ and matrix $B$ should be $3000 \times 1000$, and at the beginning input values for $n$ ($\leq 6000$), $m$ ($\leq 3000$) and $p$ ($\leq 1000$).

Initialize members of matrix $A$, as follows $a[i][j] = i \times j$, for $i=0, 1, 2 \ldots n-1$, $j=0, 1, 2, \ldots, m-1$. Initialize members of matrix $B$, as follows $b[i][j] = i+j$, for $i=0, 1, 2 \ldots m-1$, $j=0, 1, 2, \ldots, p-1$.

1. The initial (i.e. starting) thread of your program should first do multiplication of those two matrices according to the algorithm given above. Let matrix $C_1$ gets result of this multiplication, and a time needed to perform calculation is displayed. Do not include time for initialization of matrixes $A$ and $B$. Note: For getting times I have used gettimeofday function.

2. Next, your program should do multiplication according to the algorithm that decomposes the given algorithm such that 2 threads could work in parallel. Your initial thread (the one that performed calculations in Step 1) creates 2 threads that will do appropriate calculation and each thread terminates when done with its task. **Both new threads should be executing the same function.** Let matrix $C$ gets results of this calculation and a time needed to perform the calculation is displayed. If the system has at least 2 processors, you should see improvements in time. Then, compare matrix $C$ with matrix $C_1$, and indicate if there are any differences; if your solutions are good there will be no differences.
3. Next, your program should do multiplication according to the algorithm that decomposes the given algorithm such that 3 threads could work in parallel. Your initial thread should initialize all integers in matrix C to zero and then it should create 3 threads that will do calculation and each thread terminates when done with its task. **All new threads should be executing the same function as two threads executed in Step 2.** Let matrix C gets results of this calculation and a time needed to perform the calculation is displayed. If the system has at least 3 processors, you should see improvements in time. Then, compare matrix C with matrix C1, and indicate if there are any differences; if your solutions are good there will be no differences.

4. Repeat calculation with 4 threads, then with 5 threads, and finally with 6 threads similarly as described in Steps 2 and 3 above

**On/before due day/time:**

1. electronically submit your source code on Carmen Cse2431 at Assignment Lab5 and
2. electronically submit the output of your program for:
   a. n= 2100, m= 1500 & p= 1000
   b. n= 6000, m= 3000 & p= 1000

**10-25-2019 1:30pm Your output could look like this:**

```
[babic.1@cse-fac1 Thread.Matrix]$ xNew2019 6
Enter n (<=6000), m (<=3000), p (<=1000): 6000 3000 1000

n= 6000, m= 3000, p= 1000
Threads Seconds
1 266.79
2 132.48
Comparison: No error

3 88.72
Comparison: No error

4 70.58
Comparison: No error

5 73.25
Comparison: No error

6 75.27
Comparison: No error
```
Enter n (<=6000), m (<=3000), p (<=1000):  2100 1500 1000

<table>
<thead>
<tr>
<th>Threads</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>47.67</td>
</tr>
<tr>
<td>2</td>
<td>23.56</td>
</tr>
<tr>
<td>3</td>
<td>15.77</td>
</tr>
<tr>
<td>4</td>
<td>12.10</td>
</tr>
<tr>
<td>5</td>
<td>14.60</td>
</tr>
<tr>
<td>6</td>
<td>12.29</td>
</tr>
</tbody>
</table>

Comparison: No error