CS6560 JAN-MAY 2018
Teacher: Prof. Madhu Mutyam
Assignment 3
OpenMP Parallel Programming
Due Date: -

1. **Introduction**

   You are required to write parallel program solutions for the below-given problems. Your code will be tested for both correctness and performance. Your code should be free from any data-races (even benign). This is **not a group** assignment. Your submissions will be checked for plagiarism.

   You can make use of the following information of the test system to fine-tune your code's performance.

   Details of test system:

   CPU: 4 physical core with hyper threading. (Core ids 0-7).
   Default number of Threads: 8
   Default scheduling: Static
   Default Nested Parallelism: False
   Default Data status: Shared
   Default Dynamic/Guided chunkSize: 1
   Default compiler Optimization flag: -O3

   You are free to change any of these programmable values to improve your code performance.

2. **OpenMP Parallelization**

Please see the c program attached. Parallelize the kernel using OpenMP constructs optimize for performance. The value of N is not fixed but you can assume that N is going to be a multiple of 8. Any edits outside the marked region will be discarded. While submitting, rename the file with **your roll number** (in lower case).

Evaluation Criteria:

Minimum Criteria for evaluation: Correctness
Best Performance: 40% (For num_threads = 8)
Parallelism: 40% (scale up)
Lack of Races: 20%

```
/*  EDIT BELOW THIS LINE */
for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            for(k=0;k<N;k++)
                c[j][i] = c[j][i] + a[k][i]*b[k][j];
```

```
 for(i=0;i<N;i++)
         for(j=0;j<N;j++)
             vec[j] += c[i][j] - a[i][j] - b[i][j];
/* EDIT ABOVE THIS LINE */
```

The execution format
$ ./a.out <no of threads>

3. **Analyzing parallel program performances.**
   Consider the following two parallel programs. Report the execution time of both programs for different number of parallel threads (1, 2, 3, 4) . *The test computer should have at least 4 hardware threads available.* Try reasoning your observations. Choose considerably large values for N (in millions) if memory permits. Run the programs separately to remove chances of varying cache performance.

Also try following configurations:

i. Dynamic / guided scheduling for different chunk sizes.

```
Program A:
#pragma omp parallel for schedule(static)
for(i=0; i< N; i++ ) {
        if(i%2 == 0)
      a[i] = b[i] - c[i];
   else
      a[i] = b[i] + c[i];
      }
```

Program B:

```
#pragma omp parallel for schedule(static)
for(i=0; i< N; i++ ) {
        if(i%2 == 0)
       a[i] = b[i] - c[i];
 }
```

4. **Submission Rules**
   You have to submit a tar file with following name: *rollNumber.tar.gz*. The folder should contain a makefile, which will build the programs when we type *make* in the folder name *rollNumber***.** The submission should also contain *rollnumber_report.pdf* and updated C file *rollnumber.c*.