Assignment One: Introduction

In this assignment, you will learn how to use tools to improve your productivity. To make the assignment meaningful, we will look at the use of fixed point arithmetic in the context of matrix multiplication. You will first document how performance scales with matrix size. Subsequently, you will implement Matrix Multiplication in fixed point to (potentially) improve the performance.

Please do the steps mentioned in this link to understand the difference between Fixed and Floating Point

https://in.mathworks.com/help/fixedpoint/ug/visualize-differences-between-floating-point-and-fixed-point-results-app.html

This will help you understand the nuances of this area.
There is another link that is very useful which enunciates the difference.

To measure time, you can use this microsecond timer code:

```
#include <sys/time.h>
#include <unistd.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

const double micro = 1.0e-6;
double calculateTime()
{
    struct timeval TV;
    struct timezone TZ;

    int RC = gettimeofday(&TV,&TZ);

    if (RC == -1){
       printf("Bad call to gettimeofday\n");
       return(-1);
    }

    return ( ((double)TV.tv_sec ) + micro * ((double)  TV.tv_usec
));

}
```

Your matrixMul.c should have the following structure and implement the functions with appropriate arguments:

```
void printMatrix(…){
   //Helper to print the Matrix passed as argument
}
int main(int argc, char **argv)
{
// Process command line arguments
// n, # repetitions, block size

// Start the timer
for (r=0; r<nreps; r++){ //For statistical correctness
    for (i=0; i<n; i++)
```

```
            for (j=0; j<n; j++){
                sum = 0;
                for (k=0; k<n; k++)
                        sum+= A[i][k] * B[k][j];
                C[i][j] = sum;
            }
    }
    // Stop the timer
            calculateTime()
    //The plotting can either be done in this place or you can save the times for different matrix sizes
    and gnuplot can be called externally.
    }
```

What do you have to do:

- Go through the mathworks link and the other link thoroughly
- Get matrix Multiplication (on random input values) working
    - Use make and project structure as discussed
    - Vary n – the size of the matrix
    - Log the runtimes
    - Plot the graph using gnuplot
- Implement fixed point arithmetic (basically addition and multiplication)
    - 16/32 bit word
        - Q8/Q12/Q16/Q24
    - Repeat steps of the previous bullet using Fixed point

What do you have to turn in:

- Document your work in a well-written report (pdf – no other extensions must be present in the Project directory) which discusses your findings carefully. Provide all data you plotted in tabular form. This should also have the following (obtained by executing the associated command). Place this in a file called env.txt in your Project directory.
    - OS name : uname -a
    - Your Machine : hostname
    - Compiler version number: gcc -v
- gzip the Project directory and submit in Moodle. Please ensure that you run Make clean before you run gzip/tar -zxvf.
- Demo:
    - The graph must be generated automatically by typing make.