Lab One: Familiarization with System Tools

Major learnings: Get familiar with lex, yacc, Gnuplot, [svn](https://moodle.iitd.ac.in/mod/url/view.php?id=45500" \o "SVN), [make](https://moodle.iitd.ac.in/mod/url/view.php?id=45501).

Background:  Real numbers define the real world. Unfortunately, all machines have a finite word length and hence we use approximations. Most modern machines use Floating-point representation to represent real numbers. However, they are expensive in terms of the number of CPU cycles as well as power requirements. The alternative used in many settings, in particular, in signal processing, is to use fixed-point arithmetic. Fixed point arithmetic is much faster than floating-point arithmetic. In your computer architecture classes, you would have studied the same.

Deliverables:

1. You will build a library to implement 32-bit fixed-point arithmetic (which would be parameterizable). The arithmetic operations you need to implement in the library are addition, subtraction, multiplication, and exponentiation.
2. You have to use [lex and yacc](https://moodle.iitd.ac.in/mod/resource/view.php?id=45499) to generate the tokens (parsing the input files) and implement the grammar for the library. Use Input file names as matrix1.txt, matrix2.txt. You need to use the library to implement the matrix operation (multiplication and inverse).
3. Follow standard project directory structure for the submission, i.e. /src/ : source files, /lib/ : library.
4. Write [Makefile](https://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/) for your project.
5. Zip the project and rename to ENTRYNUMBER\_FIRSTNAME\_SECONDNAME.zip
6. You should use [svn](https://help.ubuntu.com/community/Subversion) to manage the commit versions of your project. Submit history of changes from[svn](https://moodle.iitd.ac.in/mod/url/view.php?id=45500) log on every submission

Document the performance gain (time vs. size) of the fixed-point over floating-point implementations. Visualize it using Gnuplot. Size: matrix size variation

Document the precision vs. size for the same. Visualize it using Gnuplot.

Note: While comparing the fixed-point to floating-point, you must disable Vectorization  to not use SSE/SSE2 instruction.

Pointer to FixedPoint Arithmetic:

<https://courses.cs.washington.edu/courses/cse467/08au/labs/l5/fp.pdf>

Use "build" to compile your code and "run" to execute (in makefile)

We will run these commands to run your code

* Multiplication :- [make](https://moodle.iitd.ac.in/mod/url/view.php?id=45501) run Op=1 m1 = "matrix1.txt" m2 ="matrix2.txt"
* Inverse : - [make](https://moodle.iitd.ac.in/mod/url/view.php?id=45501) run Op=2 m1 = "matrix1.txt"
* Exponentiation :- [make](https://moodle.iitd.ac.in/mod/url/view.php?id=45501) run Op=3 m1 = "input.txt"

input.txt will have 2 space separated numbers

sample input file :- [matrix1.txt](https://moodle.iitd.ac.in/pluginfile.php/213075/mod_assign/intro/matrix1.txt)