Software Requirements Specification
For
Enumeration Mathematical Library

Instructor :	David Augenblick
Team Members :	Mosfiqur Rahman (Project Manager), Debdut Karmakar(Developer**)**
Cycle :	7
Date Submitted :	January 27, 2016.

1 Introduction

"Enumeration Mathematical Library" or EML will be a fast C++/C library for mathematical computations. It will concentrate in number theory and support from basic number theory to power series, polynomials, p-adic fields, and algebraic and combinatorial number theory.

In addition, EML will provide various low-level routines for fast arithmetic. It will be extensively documented and tested.

We are planning to write it in ANSI C, which will run in many platforms.

1.1 Scope

"Enumeration Mathematical Library" will support :

- Arbitrary Large Integers
- Arbitrary Precision Numbers
- Basic Number Theoretic Functions
- Complex, Quaternions and Higher Fields
- P-adic Fields
- Algebraic Number Theory
- Combinatorial Objects and Functions
- Matrix Algebra

Currently, supported architectures include x86-64. There is also support for parallel programming.

1.2 Definitions, Acronyms, and Abbreviations

ANSI :	American National Standard Institute
EML :	Enumeration Mathematical Library
x86-64 :	64 bit version x86 instruction set
MPI :	Message passing interface

1.3 User Profile

EML will be primarily used in scientific computing and mathematical research. Its support for parallel computing will make it suitable for cluster computing. Although the main target applications for EML are number -theoretic and other mathematical research, students are also a major target and even it will be used in cryptography and security systems. It can be used by engineers for solving complex problems too, and doing fast calculations.

2 External Interface

EML is a C++/C library that can be accessed through its header files.

2.1 User Interface

In C programs, EML library can be accessed by including the header files. For example, to include arbitrary precision library from EML, #include <eml/arb.h>. Then while compiling use the proper flags for EML like -leml in GCC.

For C++, you need to #include <eml/arb>. Then while compiling use the proper flags for EML like -leml in GCC.

2.2 Data Interface

EML is generally stand-alone and it does not require any data interface in general cases. However, in case of cluster computing, data interface is required for data transfer between different nodes. OpenMP and MPI libraries handle it.

3 Specific Requirements

3.1 Functional Requirements

The statements below define the functional requirements for the system.

001 - Arbitrary Large Integers

The library will have a separate package for arbitrary large integers and possible operations with it.

002 - Arbitrary Precision Numbers

The library will have a separate package for arbitrary precision floating point numbers and operations on it.

003 - Complex, Quaternions and Higher Fields

The library will have a package for different fields starting with complex numbers and quaternions, p-adic fields. It will include axioms and operations for those fields.

004 - Combinatorial Objects

The library will support combinatorial objects like combinations, permutations, power series, recurrences, tabuleax, q-analogues, etc.

005 - Graphical Library

It will support a graphical library for plotting, drawing mathematical objects and other graphical renderings and data.

3.2 Performance Requirements

The statements below define the functional requirements for the system.

001 - Fast Library

EML will be carefully designed to be as fast as possible, both for small operands and for huge operands. The speed is achieved by using full words as the basic arithmetic type, by using fast algorithms, with highly optimised C code for the most common inner loops for a lot of CPUs, and by a general emphasis on speed.

002 - Fast Algorithms

The mathematical algorithms will be as fast as possible.

003 - Parallel Programming

The library will support parallel programming by using OpenMP and MPI libraries.

3.3 Design Constraints

3.3.1 Writing Fast Mathematical Algorithm

Reason : Writing and gathering fast mathematical algorithms is time consuming in general.

3.3.2 Fast Graphical Rendering

Reason : In general, graphical renderings are not fast and resource consuming. Current graphics packages and libraries are not enough fast and needs to be re-written.

3.4 Data Requirements

Our data requirements include thousands of variable, each of which is impossible to specify concerning the time constraint. Still we provide some of our data requirements just for example.

001 - Complex Library

Name	Data	Type	Comments
Complex Size	complex_size	size_t	Stores size of complex number
Complex Argument	complex_arg	double	Stores the argument of complex number

Imaginary Part	complex_img	double	Stores the imaginary part of complex number
Real Part	complex_real	double	Stores the real part of complex number
Modulus	complex_mod	double	Stores the modulus of complex number