Google Summer of Code Proposal for Boost.Multiprecision project

# Personal Details

* **Name:** Jyotesh Rajesh Choudhari
* **College/University:** Indian Institute of Technology Bombay
* **Course/Major:** Computer Science and Technology
* **Degree Program:** Master of Technology (M.Tech.)
* **Email:** [jyoteshrc@gmail.com](mailto:jyoteshrc@gmail.com)
* **Availability:**
  + **Time to spend on your GSoC:**

I will spend 30 to 35 hours per week on my GSoC project.

* + **Intended start and end dates:**

My intended start and end dates are as per the GSoC schedule i.e. June 17th 2013 and September 16th 2013 respectively. But it may vary according to mentor’s advice.

* + **Factors affecting my availability:**

I will not be taking any courses, but my M.Tech. research project will start in the month of June. I will be able to give enough time for GSoC.

# Background Information

**Educational Background:**

I completed by Bachelor’s degree in Computer Science and Technology from Visvesvaraya National Institute of Technology in 2011. The courses which I had taken are Object Oriented Programming, Data Structures and Program Design, Analysis of Algorithms, Operating Systems, Computer Architecture, Concepts in Programming Languages, Numerical Computing, System Programming, Theory of Computation, Software Engineering, Compilers, Database Management Systems, Computer Networks, Artificial Intelligence, etc.

In 2011, I joined Master’s program in Computer Science and Technology as a Research Assistant at Indian Institute of Technology Bombay. My research field is Machine Translation. I work as a System Administrator in part-time. I have taken courses like Artificial Intelligence, Machine Learning, Information Retrieval, Natural Language Processing, Probabilistic Graphical Models, Statistical Relational Learning, etc.

**Programming Background:**

This is the first time I will get involved in a large open source project. I have neither done any internship before nor worked on any large OSS project.

Some of the course assignments I have done are:

* Implementation of generic wrapper functions which use function pointers and void pointers to handle linked lists of integers, floating point numbers and structures and to provide operations on them (used C)
* Conversion of a small set of assembly language instructions into their equivalent binary and hex forms (used C++)
* Implementation of very small banking application to understand the use of OOP features of C++ like inheritance, friend class, etc. (used C++)
* Back Propagation algorithm for training neural network (used C++)
* A simple digit recognition application using neural network (used C++)
* A\* algorithm to solve Blocks World problem (used Prolog)
* English Part Of Speech Tagger using Hidden Markov Models (used Python)
* Context Free Grammar for Verb Group Detection (used Lex and Yacc)
* Implementation of Decision Tree Classifier, Naïve Bayes Classifier and K-Means Clustering algorithm (used Java)
* Message Passing Algorithm on undirected graphical models (used Java)
* Index Tuning Wizard for PostgreSQL to find which indices will best benefit a given workload of queries (This required a little modification of PostgreSQL code) (used C and Java)
* Simulation of 2D and 3D graphics pipelines (used C++ and OpenGL)
* Animating a car (used C++ and OpenGL)

**Programming Interests:**

I love programming. I practise on online judges like [SPOJ](http://www.spoj.com/users/jyoteshrc/) and [CodeChef](http://ww2.codechef.com/users/jyotesh) to improve my programming skills and algorithmic knowledge. Though the programs which I normally write in C++ do not involve much OOP concepts, I am very much interested in these as things like code reusability and abstractions are very important for development of large projects.

**Interest in contributing to the Boost C++ Libraries:**

Boost is a very powerful, rich and efficient library for C++. It is widely used by programmers. It is an honor to contribute to Boost library. Working on this library will help me in improving my programming skills. It will also give me a chance to communicate with good programmers all around the world. Boost will serve as a platform for me to enter into open source software development.

**Interest in the project I am proposing:**

I have a lot of interest in algorithms and data structures. This project will involve the implementation of algorithms and data structures. This motivates me to take this project. I have implemented basic algebraic operations like addition and subtraction for fixed length large radix-10 integers (containing 1000 or more digits) in C for some of the problems on SPOJ. I always wanted to implement algorithms like Karatsuba algorithm or FFT-based multiplication algorithms and read the material on these, but never implemented these. I also wished to write a class for large floating-point numbers in C++ similar to BigDecimal in Java, so that I could use the code in future. In the project on Boost.Multiprecision, I will get to develop a radix-2 multiprecision floating point number class which will be efficient and rich in functionality and it will not only be used by me, but also by whole world. This further motivates me to take this project.

**Previous work in this area before or on similar projects:**

As I mentioned above, I have written codes containing large integers for some problems on SPOJ in C. Those problems required only addition, subtraction, multiplication of a large integer with a small integer (by small I mean which usually fits in integer data type) and division of a large integer with a small integer. This will help me to some extent while developing the floating point numbers class.

**Plans beyond Summer of Code time frame for my proposed work:**

I have not contributed to any open source project before. Boost will be my first open source project. I will gain a lot of experience after this project. I will utilize my experience to continue to work on Boost libraries and other OSS projects.

**Knowledge on the following languages, technologies or tools:**

* C++ 3
* C++ Standard Library 3
* Boost C++ Libraries 1
* Subversion 2

**Familiarity with software development environments:**

I am familiar with Visual Studio and Netbeans. I am also comfortable with programming using terminal (using g++) in linux.

**Familiarity with software documentation tools:**

I have not used any software documentation tools till now, but I will learn Doxygen or any other software documentation tool before June.

# Project Proposal

**Description of my proposed work:**

Boost.Multiprecision is a new library which offers multiple precision floating-point types with precision higher than those of built-in float, double and long double. The task is to develop a high-performance back-end for radix-2 floating-point numbers. Currently <boost/multiprecision/cpp\_dec\_float.cpp> provides radix-10 floating-point numbers which has certain performance losses and lack of extensibility. The new radix-2 implementation is expected to give high performance. The radix-2 implementation will provide a more natural conversion to and from C++ built-in floating-point types having radix-2 such as float, double and long double.

High precision floating-point numbers are required in scientific and mathematical applications. Some languages provide high precision floating-point numbers such as Java (java.math.BigDecimal) and Python (decimal.Decimal). For C, GNU MPFR Library provides multiple precision floating-point numbers, whereas for C++, GNU MP Bignum Library provides such facility. Boost library also has back-ends using above two libraries, viz. <boost/multiprecision/gmp.hpp> and <boost/multiprecision/mpfr.hpp>, which provide very fast and efficient support for multiple precision floating-point numbers. The aim of this project is to develop a very efficient radix-2 floating-point number back-end which does not depend on any other library.

Initial sketch is present at sandbox/big\_number/boost/multiprecision/cpp\_bin\_float.hpp. Class cpp\_bin\_float is a template class in namespace boost::multiprecision with parameters like exponent type, allocator to be used, short limb type, long limb type, etc. This class will provide the facility to use allocator of programmer’s own choice. If an allocator is not specified, array will be taken as a default allocator. The class will provide various constructors to initialize the member variables from character-based string, unsigned integer, signed integer and inbuilt floating-point type variables (float, double and long double). The class will also provide copy constructors. The details of these functions are yet to be fully implemented. This class will be developed further. The main steps in development are:

* Provide I/O handling for character-based strings and C++ <iostream> and <iomanip> support.
* Implementing basic algebraic operations i.e. addition, subtraction, multiplication and division.

Multiplication will be implemented using Karatsuba algorithm which is an efficient algorithm with O (nlog23) complexity. Optionally for larger numbers (more than 10000 decimal digits), FFT-based algorithm like Schönage-Strassen algorithm can be implemented which has O (n log n log log n) complexity.

Division will be implemented using ShortDivision or DivideNewton algorithm.

* Support for the elementary transcendental functions listed in <cmath>.
* Seamless interoperation with Boost.Math.

# Proposed Milestones and Schedule

**Estimated milestones and schedule:**

* Till June 16th:
  + Read more about Boost.Multiprecision, algorithms to be used in the implementation, STL library
  + Learn Doxygen and similar tools for documentation
  + Discuss with mentor about implementation details
* June 17th - June 26th:
  + Provide I/O handling for character-based strings
  + Provide C++ <iostream> and <iomanip> support
* June 27th - July 1st:
  + Write unit tests to check I/O support
* July 2nd - July 11th:
  + Discuss with mentor about implementation of addition and subtraction
  + Code up addition and subtraction methods
* July 12th - July 16th:
  + Write unit tests to check correctness of addition and subtraction
* July 17th - July 28th:
  + Discuss with mentor about multiplication and division algorithms to be used
  + Code up multiplication and division methods
* July 29th - August 1st:
  + Write unit tests to check correctness of multiplication and division
* August 2nd - August 26th:
  + Discuss with mentor and code up transcendental functions
* August 27th - August 31st:
  + Write unit tests to check correctness of functions
* September 1st - September 16th:
  + Discuss with mentor and integrate with Boost.Math
  + Write unit tests
* September 17th - September 22nd:
  + Clean up and reorganize code
  + Write more tests for thorough checking
  + Write documentation