# Google Summer of Code 2013 Proposal

Boost.org – Boost.MultiPrecision

## Personal Details

| | |
|---|---|
| Name | Saurav Bhattacharya |
| College | Indian Institute of Technology Kharagpur |
| Course | Computer Science and Engineering |
| Degree | Bachelor of Technology (honors) in Computer Science and Engineering *and* Master of Technology in Computer and Information Technology |
| Email | online.saurav@gmail.com |
| Availability | I plan to spend about 40 hours in a week for my GSoC 2013 project<br>My intended start date is June 1st 2013 and end date is September 15th 2013<br>My academic semester after the university summer vacation starts roughly from the 3rd week of July |

## Background Information

### Educational Background

I am currently enrolled in the Dual Degree Course in the Department of CSE at IIT Kharagpur. As part of my curriculum I have taken the following courses (chronologically):

- Physics*
- Programming and Data Structures*
- Mathematics I and II
- Electrical Technology*
- Algorithms I and II*
- Discrete Structures
- Introduction to Electronics*
- Formal Language and Automata Theory
- Switching Circuits and Logic Design*
- Probability and Statistics
- Computer Organization and Architecture*
- Compilers*
- Artificial Intelligence
- Database Management Systems*
- Operating Systems*
- Complex Network Theory
- Theory of Computation
- Computer Networks*
- Cryptography and Network Security
- Logic for Computer Science
- Machine Learning
- Distributed Systems

** contains additional lab component*

## Programming Background

My internships are summarized as follows (reverse chronological order):

- Google Summer of Code 2012 at European Smalltalk User Group
  - Pharo Smalltalk
  - Microsoft Excel XLSX Format
  - May 2012 to August 2012

- École Polytechnique Fédérale de Lausanne
  - Java Standard Edition
  - Swing GUI Toolkit
  - Java Universal Network Framework
  - Scripting in AWK and Bash
  - PostgreSQL database management
  - May 2011 to July 2011

- IIM Bangalore
  - Adobe Flex
  - ActionScript
  - MySQL database management
  - May 2010 to July 2010

My Open Source contribution is limited to my work at ESUG, but I want to do more in this area. Boost.org will provide a great window of opportunity for me to start contributing to Open Source. This is one of the reasons why I am applying to Boost.org for my Google Summer of Code 2013 project.

As part of my curriculum, I have also written various programs in C++, Java and Python. They are summarized as follows:

- 64 bit pipelined MIPS-like processor
- Compiler for a *Reduced C* language
- Software tool for ranking of courses in a University
- UNIX-like Memory resident file system
- Text-based conferencing system for Linux

Last, but not the least, I have written some programs *just for fun.* Some of them are:

- A Simple To Do list
- Debt Managing Software for monitoring money lent to/borrowed from friends
- Visualizer tool for displaying players statistics (in a local competition)
- Estimator for the number of Domino's Pizza Stores in India (original idea was based on a published paper)

## Programming Interests

Programming is an activity which is part of a *very* complicated system invented by man and yet it is easy enough to be done by a 4th grader. When looking at the big picture it never fails to amaze me and I rate programming as high in my interests as I rate Music.

C++, much different from how I understand the language *now*, was the first *real* programming language I worked with. The amazing ways C++ drives technology, the entire economy and the whole world. Boost Libraries, ones that make C++ even more powerful to use and work with. It is an honor for me to be even be asked to write a reason for having an interest in contributing to Boost C++ Libraries.

## Proposed Project Interests

Of the various project ideas mentioned in the ideas' page, Boost.MultiPrecision seemed most intuitive and clear as to what the project is about, and what is expected of me as a student. There were a couple of other ideas too, but on reading up further on some of them I finally narrowed down to my current choice due to the following points:

- It is an algorithmic intensive project, something which always interests me. Such projects provide room for innovating and trying new and better methods and technologies, and I like that.
- The guideline for the project steps have already been provided in the website. So instead of spending more time figuring out the smaller chunks, I can directly use those guidelines and use them as a starting place to more efficiently plan my timeline.
- The project focusses on C++, Templates and the STL: something that I have been wanting to work on. Through GSoC, this will be an excellent opportunity for me to learn and do more in this area.
- It is an optimization problem. The current implementation of Boost.MultiPrecision suffers from performance losses and lack of extensibility. This project includes writing a radix-2 floating-point back-end to improve performance, and provide more extensibility in the process. In layman's terms, making something existing, better: that is something which has always interested me.

## Relevant Previous Work

I have written code for various algorithms both in my curriculum as also during my internships. Some of them are:

- Merge Sort
- Dijkstra's
- Gradient descent
- Ford-Fulkerson
- Girvan Newman
- Jarvis March
- Round Robin
- Data Encryption Standard

All my academic projects were written in C/C++ (without using C++ exclusive features – OOP, STL, Templates). I have also worked intensively on register level code in Verilog (CPU Design and Encryption cipher) and that gave me insight into the bit level nuances of programing, something that will prove to be very valuable for this project

### Plans Beyond

As I have mentioned earlier too, I wish to continue contributing to Boost C++ Libraries and be a part of the Open Source Movement. I have great reverence for the OSS Ideology and Boost C++ will provide me with a golden opportunity in this regard. Having completed my GSoC project, I will be at a better position to re-evaluate my contributing mode and strategy. I will continue to maintain my project and take it further; I will even strive to be a mentor for a student next year.

### Contextual Knowledge

The ratings are as follows (0 to 5):

| | |
|---|---|
| C++ | 4 |
| C++ Standard Library | 3 |
| Boost C++ Libraries | 2 |
| Subversion | 2 |

### Development Environments

I am most familiar with Eclipse and Visual Studio for coding in C++. I am also comfortable with terminal based programming in UNIX like environments.

## Project Proposal

### Background

Boost.Multiprecision is a new Boost library that offers multiple precision integer, rational and floating-point types with precision exceeding those of built-in float, double and long double. Boost.Multiprecision uses a uniform architecture that embodies the extended precision type with a front-end generic number template combined with one of several back-end number types that grind out the nuts-and-bolts of the multi precision work.

The current floating-point back-end for Boost.Multiprecision uses decimal (radix-10) and suffers certain performance losses and lack of extensibility therefrom. This project will provide a binary (radix-2) floating point back-end that is expected to improve efficiency and provide a more natural conversion to and from C++ built-in floating-point types having radix-2 such as float, double, and long double.

### Related Work

There are other multi precision frameworks available for floating point data, two of which are:

- GNU MP Bignum Library
- GNU MPFR Library

Both of these use a high performance radix 2 implementation for their respective floating point backends which make both of them fast and efficient. Boost provides high performance floating point support using these external libraries:

- boost/multiprecision/gmp.hpp
- boost/multiprecision/mpfr.hpp

However, the code including the above header files becomes dependent on external libraries. Boost would like to have its own version of a self-contained binary (radix 2) floating point multi precision backend, using only C++ implementation.

## Problem Definition

The current decimal floating point backed is defined in *cpp_dec_float.hpp.*
The task for this project is to write a multi precision floating point backend using radix-2 implementation say, *cpp_bin_float.hpp*

- Class template *cpp_bin_float* fulfilling all the requirements of a Backend type (link)
- I/O handling and C++ <iostream> and <iomanip> support for the radix-2 FP-backend
- Implementation of basic algebraic functions (addition, subtraction, multiplication, and division)
    - Decision on classes of precision ranges to be supported and their respective algorithms
    - Two precision levels for Multiplication/Division
        - below THRESHOLD : Algorithm 1
        - above THRESHOLD : Algorithm 2
        - (see Resources below )

Optionally (only if time permits):

- Support of transcendental functions from <cmath>
- Seamless integration with Boost.Math

## Resources

The following are the list of resources:

- boost/multiprecision/cpp_dec_float.hpp – a useful guide
- cpp_bin_float.hpp available in boost sandbox – initial sketch
- Multiprecision documentation (link)
- Modern Computer Arithmetic (link) – algorithms (all relevant to radix-2) :
    - I/O – PrintFixed algorithm
    - FPAdd for floating point addition/subtraction
    - Sterbenz's theorem for floating point subtractive cancellation
    - Multiplication
        - Karatsuba's  algorithm
        - Complex FFT algorithm
    - Division
        - ShortDivision algorithm
        - DivideNewton algorithm

From my mentor, Christopher Kormanyos I shall receive:

- Introduction to the project and its high-level view
- Choice of algorithms/precision levels/additional functions to be supported
- Technical guidance and useful tips/tricks

## Proposed Milestones and Schedule

The suggested Google Summer of Code 2013 TimeLine is given here. Based on this, I have put together the following timeline of milestones:

### May 27 – June 1

Read up Boost.Multiprecision documentation
Understand Boost.Multiprecision backend requirements

### June 3 – June 8

Get familiar with existing code for decimal floating point backend
Discuss with mentor about the same
Review and discuss the floating point backend from sandbox

### June 10 – June 15

Discuss about different precision levels and respective algorithms
Check base conversion methods to and from radix-2 and radix-10
Finalize precision levels to support and the conversion methods to use
Begin coding I/O for character-based strings

### June 17 – June 22

Discuss possible methods for providing <iostream> and <iomanip> support
Continue coding I/O support

### June 24 – June 29

Take a break for some other commitments

### July 1 – July 5

Write tests for I/O support
Code for <iostream> and <iomanip> support
Write unit tests

### July 7 – July 12

Discuss and finalize Addition and Subtraction algorithms
Begin coding Addition
Write unit tests

### July 14 – July 19

Code Subtraction and Cancellation
Write unit tests
Discuss Multiplication algorithms for different ranges

### July 21 – July 26

Code high performance Multiplication
Write unit tests

### July 28 – August 3

Code for ultra-high performance Multiplication
Write unit tests

### August 5 – August 10

Discuss Division algorithms for different ranges
Begin coding for Division

### August 12 – August 17

Write unit tests for Division
Discuss backend interoperability with Boost.Math
Integrate Boost.Math with backend

### August 19 – August 31 (two weeks)

Complete Boost.Math integration
Buffer time to complete any previous unfinished work
If time permits, work on transcendental functions

### September 2 – September 7

Clean up code
Perform additional tests
Write documentation

Timeline still leaves enough time till the hard deadline – *September 23*