San José State University
Department of Computer Engineering

CMPE 180-92
Data Structures and Algorithms in C++
Spring 2017

Instructor: Ron Mak

Assignment #4

Assigned:	Thursday, February 16
Due:	Thursday, February 23 at 5:30 PM
CodeCheck:	http://codecheck.it/codecheck/files/17021606594zetayf9bostp8brrhgcx4gkr
Canvas:	Assignment 4. Big Pi
Points:	100

Big pi

You will compute and print the first thousand decimal places of pi. For extra credit, you will compute and print the first million decimal places of pi.

The purpose of this assignment is to give you practice downloading, configuring, building, installing, and using a C++ library. You may be asked to perform such tasks in future classes, so these are important skills to master.

The Borwein algorithm

Your program will use the algorithm first published by mathematician brothers Jonathan and Peter Borwein in 1985:

Set $a_0 = 6 - 4\sqrt{2}$ and $y_0 = \sqrt{2} - 1$. Then iterate

$$y_i = \frac{1 - \sqrt[4]{1 - y_{i-1}^4}}{1 + \sqrt[4]{1 - y_{i-1}^4}}$$

$$a_i = a_{i-1}(1 + y_i)^4 - 2^{2i+1} y_i (1 + y_i + y_i^2)$$

and the a_i will converge quartically towards $1/\pi$.

This algorithm is quartic – each iteration increases the number of correct digits by a factor of 4. See https://en.wikipedia.org/wiki/Borwein's_algorithm "Quartic algorithm (1985)".

The Multiple Precision Integers and Rationals (MPIR) library

From the introduction of the library's document:

> MPIR is a portable library written in C for arbitrary precision arithmetic on integers, rational numbers, and floating-point numbers. It aims to provide the fastest possible arithmetic for all applications that need higher precision than is directly supported by the basic C types.

Like many C++ software meant for Linux systems (and similarly for Mac OS and Windows), MPIR is distributed in source form. Download the zip file for MPIR 2.7.2 and its documentation from http://mpir.org/downloads.html. After you unzip the file, you will have many C source files and several shell scripts. You can run the scripts on a Linux or Mac system in a bash terminal window. If you are on a Windows system, you must run the scripts in the Cygwin terminal window, not the standard Windows command terminal.

Build and install the library

MPIR uses a conventional sequence of commands to build and install the library. See the "Installing MPIR" chapter of the documentation. First, you run the **`configure`** script which checks your system for necessary utility programs and generates the makefiles. Next, you run **`make`** which uses the generated makefiles to compile the C source files and build the library. Then you run **`make check`** which compiles and runs test programs to verify that you properly built the library. Finally, you run **`make install`** to install the MPIR library **`libmipr*`** (several files) and its header file **`mpir.h`** in the standard locations **`/usr/local/lib`** and **`/usr/local/include`**, respectively.

The build is straightforward on Linux and Mac OS. To compile and run your program **`BigPi.cpp`** on the command line:

```
g++ BigPi.cpp -lmpir -o BigPi
./BigPi
```

The **`-lmpir`** specifies using the MPIR library.

To compile and run **`BigPi.cpp`** in Eclipse, you must specify using the MPIR library for the project. Right-click on the project and select Properties. Under Linker, select Libraries. Add **`mpir`** to the Libraries box.

In Windows, you must do everything in the Cygwin terminal window in order to run the bash scripts. You must first download and install the utility programs that the MPIR build scripts need. These include **`make`** and the **`m4`** macro processor. You get them from the Cygwin website similarly to the way you got the GNU C++ compiler. If you are missing other utilities, **`configure`** will tell you. Then you must go back to the Cygwin website to download and install the missing utilities. You may need to do this several times before **`configure`** is finally happy.

When you run **`configure`** on Windows, you must include two arguments to tell it that you want to create a Windows DLL (dynamic-link library):

```
./configure --disable-static --enable-shared
```

See p. 13 of the documentation. Then run the **`make`** commands the same way as described above for Linux and Mac OS.

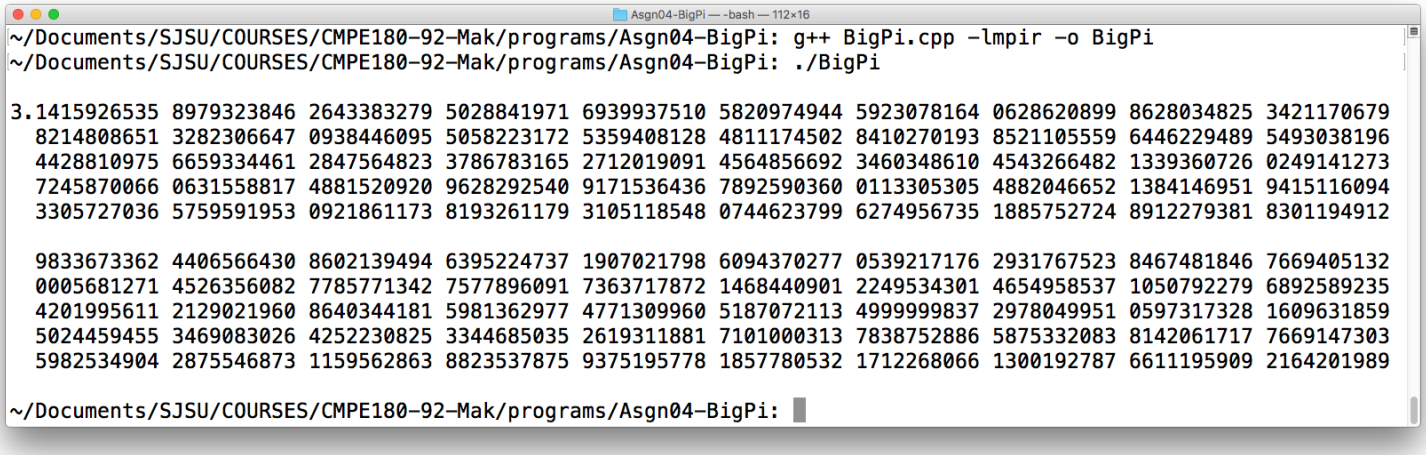
When you compile and run a program on Windows, you may also need the **`-L`** option to specify the location of the library directory:

```
g++ BigPi.cpp -L/usr/local/lib -lmpir -o BigPi
./BigPi
```

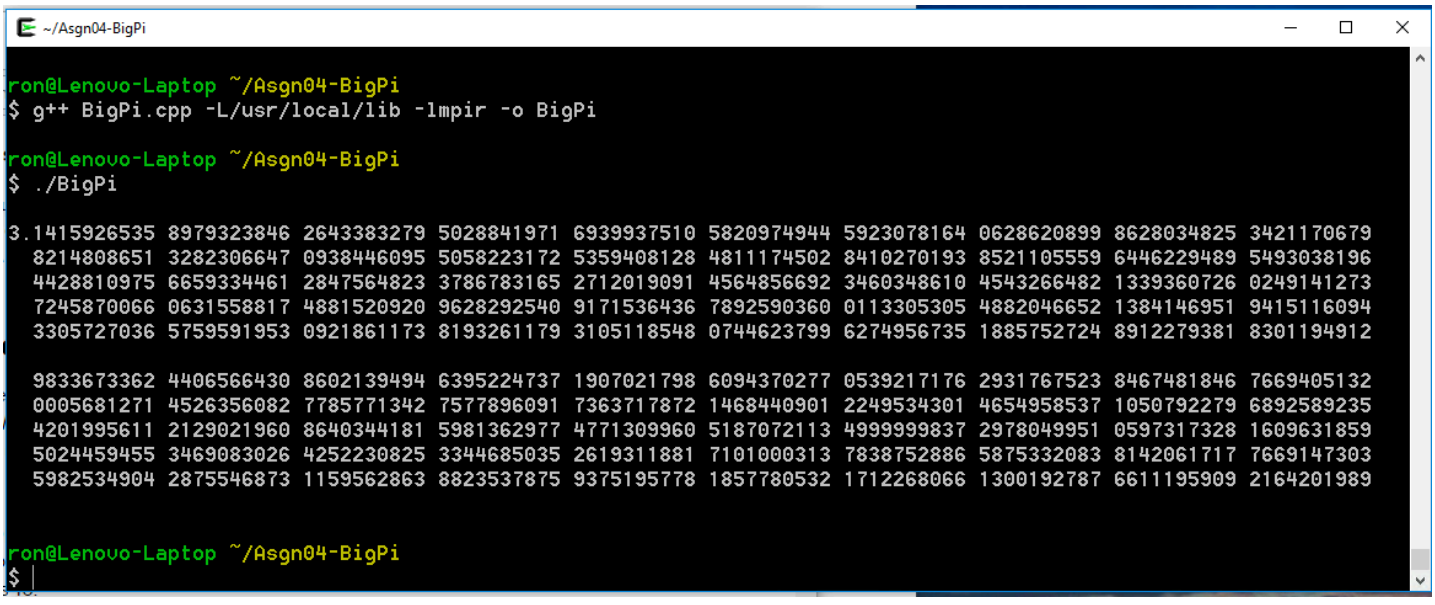
Please work together to properly configure, build, and install the MPIR library on your computers. However, as always, your program must be individual work.

Expected output

Here is a screen shot of a compile and run on a Mac:



And on Windows:



Not for CodeCheck

Because of the MPIR library, you will not be able to compile and run your program in CodeCheck. Go to the CodeCheck URL above to obtain the skeleton of your program. But otherwise you must edit, compile, and debug outside of CodeCheck.

Submission into Canvas

Submit into Canvas a copy of your C++ program and a text file of its output:
Assignment 4. Big Pi

Rubric

Your program will be graded according to these criteria:

Criteria	Maximum points
Correct output	**25**
Good use of the MPIR library • Constant values computed outside of the iterations. • Good library calls inside the iterations.	**50** • 10 • 40
Good program style • Descriptive variable names. • Good functional decomposition. • Meaningful comments. • Follow the coding style (formatting, braces, indentation, function declarations before the main, etc.) of the Savitch textbook.	**25** • 5 • 10 • 5 • 5

Extra credit (up to 25 points)

Modify your program to compute and print the first million decimal places of pi. Submit your new program and a text file of its output along with your original program and output.

To verify your results, see http://www.exploratorium.edu/pi/pi_archive/Pi10-6.html
Checking only the last few number blocks should be sufficient!

How many iterations did you need to get an accurate result? How long did it take to compute the million digits on your computer?

Academic integrity

You may study together and discuss the assignments, but what you turn in must be your individual work. Assignment submissions will be checked for plagiarism using Moss (http://theory.stanford.edu/~aiken/moss/). **Copying another student's program or sharing your program is a violation of academic integrity.** Moss is not fooled by renaming variables, reformatting source code, or re-ordering functions.

Violators of academic integrity will suffer severe sanctions, including academic probation. Students who are on academic probation are not eligible for work as instructional assistants in the university or for internships at local companies.