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, September 14
Due:	Thursday, September 21 at 5:30 PM
CodeCheck:	http://codecheck.it/codecheck/files/1709180103a6ohvwgk0u5ra3r1aahstcvcq
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 or jobs, so these are important skills to master.

The Nonic Convergence algorithm

Your program will use the **Nonic Convergence** algorithm published at https://en.wikipedia.org/wiki/Borwein's_algorithm, "Nonic convergence". Each iteration increases the number of correct digits by a factor of 9.

The Multiple Precision Integers and Rationals (MPIR) library

From the introduction in the library's documentation:

> 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 packages for Linux systems, MPIR is distributed in source form. Download the zip file for the latest version of MPIR 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 (see the next section).

The MPIR library does not include a function to compute cube roots. Therefore, you must code a “big cube roots” function that computes cube roots iteratively. See https://en.wikipedia.org/wiki/Cube_root, “Numerical methods”.

For this assignment, do not use the header file **`<mpirxx.h>`**. In other words, don’t use the overloaded arithmetic operators. Instead, call the individual functions of the API.

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 **`libmpir*`** (several files) and its header file **`mpir.h`** in the standard locations **`/usr/local/lib`** and **`/usr/local/include`**, respectively.

To compile and run your program **`BigPiNonic.cpp`** on the command line:

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

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

To compile and run **`BigPiNonic.cpp`** in Eclipse, you must specify using the MPIR library for the project. Eclipse Neon instructions:

- Right-click on the project and select Properties.
- Under C/C++ Build, select Settings.
- Under Linker, select Libraries. Add **mpir** to the Libraries box.

In Windows, you must run the bash scripts in the Cygwin terminal window. 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,, and 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++ BigPiNonic.cpp -L/usr/local/lib -lmpir -o BigPiNonic
./BigPiNonic
```

Students on Windows systems have had difficulties in past semesters compiling and installing this library. So if you're on Windows, this is a good time to consider installing VirtualBox (https://www.virtualbox.org) and a version of Linux such as Debian (https://www.debian.org) or Ubuntu (https://www.ubuntu.com).

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

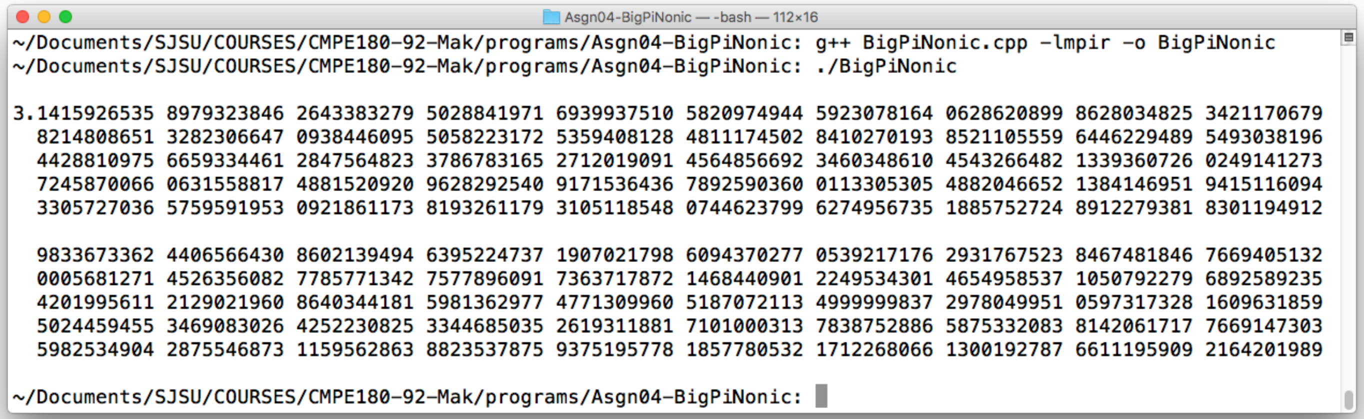
Tips

Your program should make as few iterations as possible when computing cube roots and pi. In your loops, you can compare the last two computed values and stop when they are the same. Experiment with the comparison functions **`mpf_cmp`** and **`mpf_eq`** to see which works better. But also include a hard limit on the number of iterations to prevent infinite loops.

Your program will spend much time computing cube roots, so make that function as efficient as possible. Experiment and choose between **Newton's method** or **Halley's method**. Test your cube root function separately to make sure it works! Can it compute $\sqrt[3]{125} = 5$ precisely to a thousand decimal places?

Expected output

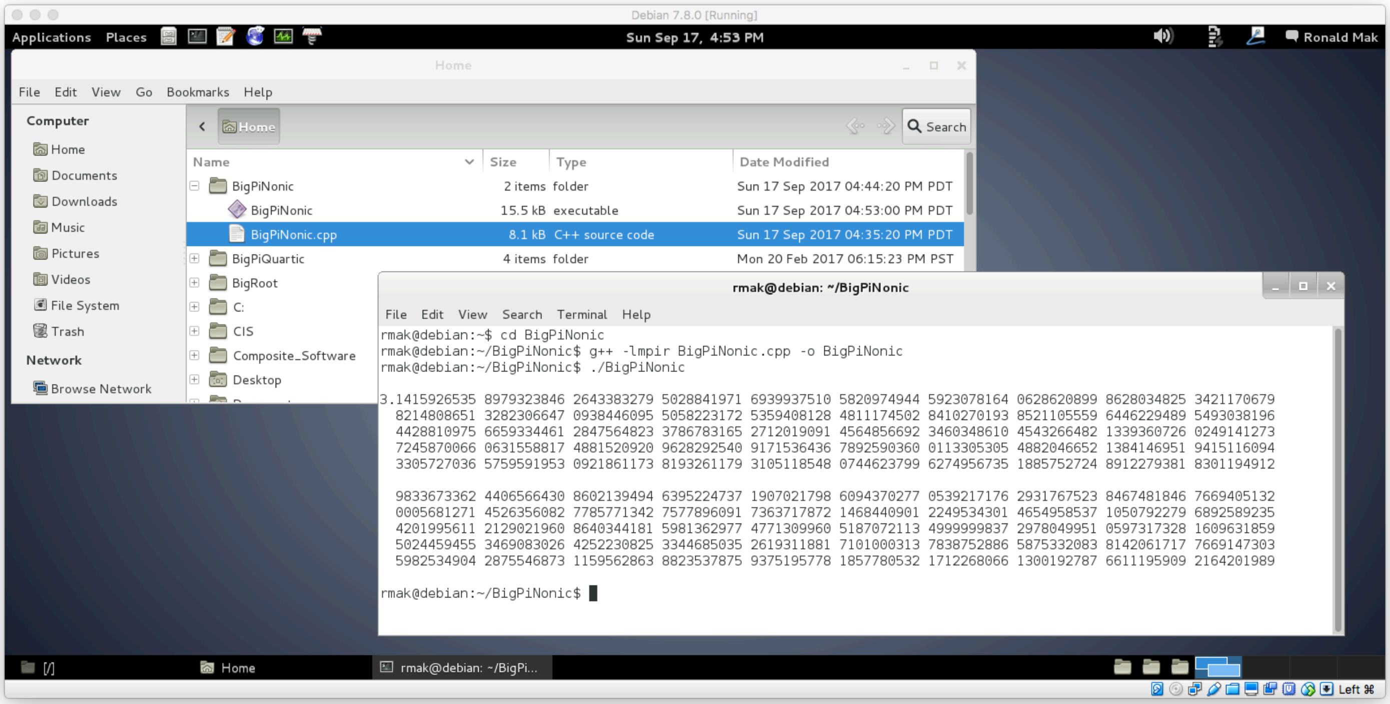
Here is a screen shot of a compile and run on a Mac:



```
~/Documents/SJSU/COURSES/CMPE180-92-Mak/programs/Asgn04-BigPiNonic: g++ BigPiNonic.cpp -lmpir -o BigPiNonic
~/Documents/SJSU/COURSES/CMPE180-92-Mak/programs/Asgn04-BigPiNonic: ./BigPiNonic

3.1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 5923078164 0628620899 8628034825 3421170679
  8214808651 3282306647 0938446095 5058223172 5359408128 4811174502 8410270193 8521105559 6446229489 5493038196
  4428810975 6659334461 2847564823 3786783165 2712019091 4564856692 3460348610 4543266482 1339360726 0249141273
  7245870066 0631558817 4881520920 9628292540 9171536436 7892590360 0113305305 4882046652 1384146951 9415116094
  3305727036 5759591953 0921861173 8193261179 3105118548 0744623799 6274956735 1885752724 8912279381 8301194912

  9833673362 4406566430 8602139494 6395224737 1907021798 6094370277 0539217176 2931767523 8467481846 7669405132
  0005681271 4526356082 7785771342 7577896091 7363717872 1468440901 2249534301 4654958537 1050792279 6892589235
  4201995611 2129021960 8640344181 5981362977 4771309960 5187072113 4999999837 2978049951 0597317328 1609631859
  5024459455 3469083026 4252230825 3344685035 2619311881 7101000313 7838752886 5875332083 8142061717 7669147303
  5982534904 2875546873 1159562863 8823537875 9375195778 1857780532 1712268066 1300192787 6611195909 2164201989

~/Documents/SJSU/COURSES/CMPE180-92-Mak/programs/Asgn04-BigPiNonic: 
```

Print the digits as shown in blocks of 10 digits and groups of 5 rows, 500 digits total per row group.

A compile and run on a Debian Linux virtual machine running on VirtualBox:


Debian 7.8.0 [Running]
Applications Places
Sun Sep 17, 4:53 PM
Ronald Mak
Home
File Edit View Go Bookmarks Help
Computer
Home
Documents
Downloads
Music
Pictures
Videos
File System
Trash
Network
Browse Network
Search
Name Size Type Date Modified
BigPiNonic 2 items folder Sun 17 Sep 2017 04:44:20 PM PDT
BigPiNonic 15.5 kB executable Sun 17 Sep 2017 04:53:00 PM PDT
BigPiNonic.cpp 8.1 kB C++ source code Sun 17 Sep 2017 04:35:20 PM PDT
BigPiQuartic 4 items folder Mon 20 Feb 2017 06:15:23 PM PST
BigRoot
C:
CIS
Composite_Software
Desktop
rmak@debian: ~/BigPiNonic
File Edit View Search Terminal Help
rmak@debian: ~/BigPi...
Left ⌘

```
rmak@debian:~$ cd BigPiNonic
rmak@debian:~/BigPiNonic$ g++ -lmpir BigPiNonic.cpp -o BigPiNonic
rmak@debian:~/BigPiNonic$ ./BigPiNonic

3.1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 5923078164 0628620899 8628034825 3421170679
  8214808651 3282306647 0938446095 5058223172 5359408128 4811174502 8410270193 8521105559 6446229489 5493038196
  4428810975 6659334461 2847564823 3786783165 2712019091 4564856692 3460348610 4543266482 1339360726 0249141273
  7245870066 0631558817 4881520920 9628292540 9171536436 7892590360 0113305305 4882046652 1384146951 9415116094
  3305727036 5759591953 0921861173 8193261179 3105118548 0744623799 6274956735 1885752724 8912279381 8301194912

  9833673362 4406566430 8602139494 6395224737 1907021798 6094370277 0539217176 2931767523 8467481846 7669405132
  0005681271 4526356082 7785771342 7577896091 7363717872 1468440901 2249534301 4654958537 1050792279 6892589235
  4201995611 2129021960 8640344181 5981362977 4771309960 5187072113 4999999837 2978049951 0597317328 1609631859
  5024459455 3469083026 4252230825 3344685035 2619311881 7101000313 7838752886 5875332083 8142061717 7669147303
  5982534904 2875546873 1159562863 8823537875 9375195778 1857780532 1712268066 1300192787 6611195909 2164201989

rmak@debian:~/BigPiNonic$
```

Not for CodeCheck

Because CodeCheck doesn't have the MPIR library, you will not be able to compile and run your program in CodeCheck. Go to the CodeCheck URL above to obtain a 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
Output	**40**
• Correct values.	• 30
• Correct formatting (digit blocks and line groups)	• 10
Good use of the MPIR library	**35**
• Constant values computed outside of the iterations.	• 10
• Good library calls inside the iterations.	• 25
Good program style	**25**
• Descriptive variable names.	• 5
• Good functional decomposition.	• 10
• Meaningful comments.	• 5
• Follow the coding style (formatting, braces, indentation, function declarations before the main, etc.) of the Savitch textbook.	• 5

Extra credit (up to 25 points)

Modify your program to compute and print the first million decimal places of pi. Study the C++ **chrono library** at http://www.cplusplus.com/reference/chrono/ and print the elapsed time (in seconds or milliseconds) for each iteration, and the total elapsed time. What percentage of the total time was spent computing cube roots?

If you have a slow computer, you can compute the first hundred thousand digits. Truly ancient computers can compute the first ten thousand digits.

Submit your new program and a text file of its output along with your original program and output into Canvas: **Assignment #4. Extra Credit**

To verify your results, see https://www.exploratorium.edu/pi/numbers-of-pi
Checking only the first and last ten digits ought to be sufficient!

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.