CSC 111 Assignment 2

NOTE: Your programs must compile and execute using the CLion environment in ECS 242. If you do your work on your own computer, be sure to test it in ECS 242 before you submit it.

Programming instructions

Assignment 2 consists of three parts, designed to develop skills writing loops and adapting existing C code. Each part requires a separate C program in a separate .c source file, and your final submission will consist of three syntactically correct C programs. Parts 1, 2 and 3 are worth 1, 2, 1 marks, respectively.

Part I

The first part of this assignment involves modifying an existing C program. In CourseSpaces, under “Sample C programs from lectures”, a program “FahrenheitToCelsius.c” has been posted, which generates a table of temperature conversions from Fahrenheit to Celsius. Your task is to write a program which generates the opposite conversion table from Celsius to Fahrenheit. You are encouraged to use the provided program as a starting point (but of course you can also start from scratch).

Your program will generate a set of conversions from Celsius to Fahrenheit in 5 degree increments from -40 degrees to +60 degrees Celsius. Your program must produce the following output.

```
-40.0 degs C =  -40.0 degs F
-35.0 degs C =  -31.0 degs F
-30.0 degs C =  -22.0 degs F
-25.0 degs C =  -13.0 degs F
-20.0 degs C =   -4.0 degs F
-15.0 degs C =    5.0 degs F
-10.0 degs C =   14.0 degs F
 -5.0 degs C =   23.0 degs F
  0.0 degs C =   32.0 degs F
  5.0 degs C =   41.0 degs F
 10.0 degs C =   50.0 degs F
 15.0 degs C =   59.0 degs F
 20.0 degs C =   68.0 degs F
 25.0 degs C =   77.0 degs F
 30.0 degs C =   86.0 degs F
 35.0 degs C =   95.0 degs F
 40.0 degs C =  104.0 degs F
 45.0 degs C =  113.0 degs F
 50.0 degs C =  122.0 degs F
 55.0 degs C =  131.0 degs F
 60.0 degs C =  140.0 degs F
```

For automated grading purposes, we require that you follow the specification of assignments exactly. In this case, your output is permitted to have different spacing or indentation than the output above, but any other deviation from the format above will result in a loss of marks. Your program must generate the output by performing calculations at runtime using a loop (i.e., **for** or **while** loop). If your program uses 'hard coded' output (i.e., by simply printing each line verbatim with a series of printf statements instead of using a loop), you will receive zero marks.

Part II

In CourseSpaces, under "Sample C programs from lectures," a program "A2P2Sequences.c" has been posted, which is missing code for three functions. The program has code in main(), which must be left unmodified. Your task is to add code to the ArithmeticSequence, LeonardoSequence and HarmonicSequence functions such that the resulting program produces the following output.

```
Arthmetic sequence: 17 23 29 35 41
Arthmetic sequence: 34 40 46 52 58 64 70 76
LeonardoSequence: 1 3 5 9 15 25 41
LeonardoSequence: 9 15 25 41 67 109 177
HarmonicSequence: 1.000 0.500 0.333 0.250 0.200 0.167
HarmonicSequence: 0.333 0.250 0.200 0.167 0.143 0.125
```

As for Part I, your output is permitted to have different spacing or indentation than the output above, but any other deviation from the format above will result in a loss of marks. Your program must generate the output by performing calculations at runtime using a loop (i.e., **for** or **while** loop).

For more information on these sequences refer to the following websites:

Wolfram

- http://mathworld.wolfram.com/ArithmeticSeries.html
- http://mathworld.wolfram.com/HarmonicSeries.html

Wikipedia

- http://en.wikipedia.org/wiki/Fibonacci_number
- http://en.wikipedia.org/wiki/Leonardo_number

Music

- https://en.wikipedia.org/wiki/Harmonic_series_(mathematics)
- http://en.wikipedia.org/wiki/Harmonic_series_(music)

Part III

Some arithmetic operations, such as addition or division, can be performed precisely by both humans and computers by following a simple mathematical process. More advanced operations, such as computing square roots, cannot be computed exactly in some cases. However, irrational functions such as square roots, logarithms and trigonometric functions are of critical importance in engineering. Instead of computing such functions exactly, engineers often use an *approximation* which is sufficiently accurate for the task at hand.

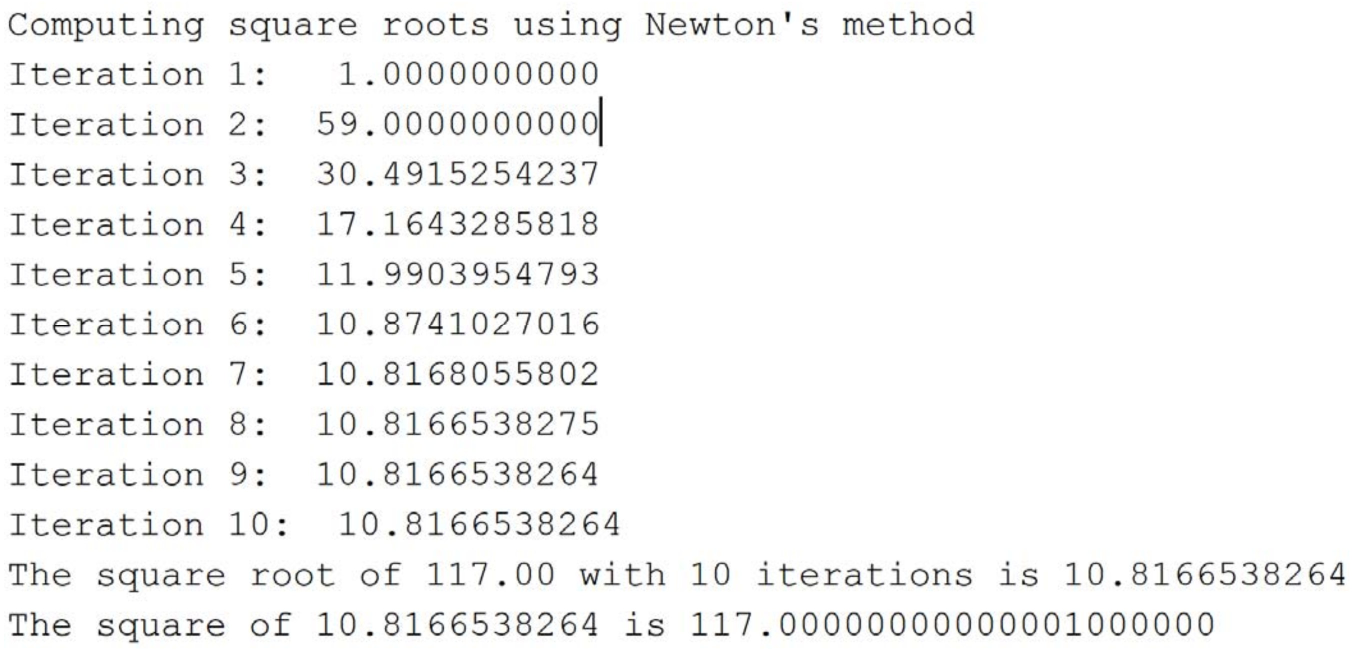
The square root button on most calculators can compute square roots to seven or eight significant digits using an approximation scheme. For this assignment, you are to use a famous approximation scheme called Newton's method—named after Isaac Newton—to approximate the value of $\sqrt{117}$.

To use Newton's method to compute the square root of a number T, the following algorithm is used. The value `count` controls the number of repetitions (and therefore the accuracy of the result).

1) Create two variables `x` and `y` of type `double`.
2) Set `x` to 1.0 (the *initial approximation*)
3) Repeat the following process for `count` iterations
   - Output the current approximation.
   - Set `y` to be the value $x - \frac{x^2 - T}{2x}$
   - Set `x` to be equal to `y`.
4) Output the final value of `x`, which will be a good approximation of $\sqrt{T}$

With each repetition of Step 3 in the algorithm above, the approximation `x` becomes more accurate. In practice, the number of repetitions is determined based on how accurate the approximation needs to be (if only a few significant digits are required, it is not necessary to have many repetitions).

Your task for Part 3 is to write a program to approximate the value $\sqrt{117}$ using the algorithm above with T set to 117 and `count` set to 10 (that is, repeating the process in step 3 a total of 10 times). Your program must use a loop (i.e., **for** or **while** loop) for the repetition. At each step, and after the loop, your program will print the value of x using the format specifier %14.10f to generate 10 decimal places and a total field width of 14 for the number of characters to output the entire floating-point number. The output of your program must be the following.



```
Computing square roots using Newton's method
Iteration 1:   1.0000000000
Iteration 2:  59.0000000000
Iteration 3:  30.4915254237
Iteration 4:  17.1643285818
Iteration 5:  11.9903954793
Iteration 6:  10.8741027016
Iteration 7:  10.8168055802
Iteration 8:  10.8166538275
Iteration 9:  10.8166538264
Iteration 10:  10.8166538264
The square root of 117.00 with 10 iterations is 10.8166538264
The square of 10.8166538264 is 117.0000000000001000000
```

Your output is permitted to have different spacing or indentation than the output above, but any other deviation from the format above will result in a loss of marks. Note that the accuracy of the approximation can be verified by computing its square. In the case of the final approximation above,

$$(10.8166538264)^2 = 117.0000000000001000000$$

which is a very accurate approximation (i.e., more accurate than most calculators).

Assignment submission instructions

CSC 111 assignments will only be accepted electronically through the assignment page on the CSC 111 CourseSpaces site. Your submission will consist of ***three C source files*** named by the following convention: If your student ID is ***V00123456,*** your C source files for parts 1, 2 and 3 (respectively) must be named **V00123456A2P1.c, V00123456A2P2.c, and V00123456A2P3.c**. In addition, your ***full name, student ID,*** and ***Assignment name*** (i.e., Assignment 2) must appear in a ***comment section at the beginning of each C program***.

For example (in Part 1):

```
Name: Polar Bear (Replace this with your name)
UVicID: V00123456 (Replace this with your student number)
Date: 2017/09/15 (Replace this with the date you wrote the program)
Assignment: A2 Part 1
File name: V00123456A2P1.c (Replace V00123456 with your student number)
Description: A Celsius-to-Fahrenheit conversion table.
```

Please submit only the source file and not the executable file. To verify that you have submitted the correct file, you are strongly encouraged to download your submissions from the site and test that they work correctly in your CLion environment. Submissions that do not follow the guidelines above will receive a mark of zero.

Since this assignment requires three source files, CourseSpaces will only allow you to submit three files. However, until the assignment due date, you may change your submission by deleting and resubmitting your source files multiple times. After the due date, no submissions will be accepted. When grading is complete, your assignment mark and comments will appear in the ***Gradebook*** section of CourseSpaces.