Scientific Computing

Assignment 3

Due: Friday, Sept. 28, 2017 at 17:00

Refer to week 1 lecture notes for instructions for submitting your assignment using the `handin` web-form. In your submission include

- all required source code and
- any other files asked for in the question.

Put the files in a directory called `lab-03`. Do not include object or executable files.

For full marks document your work using *meaningful* comments.
Make sure to include

- your name,
- your student number, and
- the assignment number.

in each file you submit.

Also add comments where necessary to clearly label each solution. **Hand in only the source files and makefile, not the objects and executables.**

---

1. (Chapter 4 – exersise 11) Create a program (`FindPrimes.cpp`) to find all the prime numbers between 1 and 100. One way to do this is to write a function that will check if a number is prime (ie. see if the number can be divided by a prime number smaller than itself) using a `vector` of primes in order (so that if the `vector` is called `primes`, `primes[0]==2, primes[1]==3, primes[2]=5`, etc.). Then write a loop that goes from 1 to 100, checks each number to see if it is a prime and stores each prime found in a `vector`. Write another loop that lists the primes you found. You might check your result by looking up a table of prime numbers. Consider 2 the first prime.

2. (Chapter 4 – exersize 18, same as Chapter 5 – exercise 7) Create a program (`SolveQuadratic.cpp`) to solve quadratic equations. A quadratic equation is of the form:

$$ax^2 + bx + c = 0 \tag{1}$$

   If you don't know the quadratic formula for solving such an expression, do some research. Remember, researhing how to solve a problem is often necessary before a programmer can teach the computer how to solve it. Use `double`s for the user inputs for `a`, `b`, and `c`. Since there are two solutions to a quadratic equation, output both `x1` and`x2`. You are only required to get it to work for real roots – for bonus points, also make it work for complex roots.

3. (Chapter 5 – exercises 2 and 4). The following program takes in a temperature value in Celsius and converts it to Kelvin. This code has many errors in it. Find the errors, and correct the code.

```
double ctok( double c) { // converts Celsius to Kelvin
  int k = c + 273.15;
  return int
}
int main() {
  double c = 0; // declare input variable
  cin >> d;     // retrieve temperature input variable
  double k = ctok("c"); // convert temperature
  Cout << k << '/n'; // print out temperature
}
```

   Absolute zero is the lowest temperature that can be reached (-273.15 Celsius). Place a check to handle bad inputs inside the `ctok` function.

4. Provide a makefile that compiles the programs in this assignment.