Stevens Institute of Technology **Assignment 1**
FE 522 – C++ Programming in Finance **Due Date: March 13, 2020**

For every problem below, create a different project folder. You should then put all these folders inside a .zip file with your name before submitting everything in Canvas. Remember to delete the ***cmake-build-debug*** *folders before doing so. This .zip file should be accompanied by a .pdf file containing a report (one single report for the whole assignment). We do not provide test cases for any of the problems, and these* ***must*** *be provided by you. When providing a solution to a problem, you must be able to present test cases alongside it,* ***otherwise it will not be accepted as a valid solution****.*

If a problem requires input and/or output files, please provide the whole file content (if not large) in the body of the report. If the file is too large to be pleasantly presented in a report (larger than a page), please provide a sample. You should include these files in folders named "input" and "output", respectively, in the root folder of each project. In order for your code to work on any machine (not only yours), use relative paths to these files in your source code:

- *for input files, use:* `"../../input/filename.txt"`
- *for output files, use:* `"../../output/filename.txt"`

Problem 1 (20 points). Study the documentation in `http://en.cppreference.com/w/cpp/numeric/random`. Choose 5 different random number distributions, generate a sample of $10,000$ numbers with each of them, and create a table (output to a file) with their mean, median, and standard deviation.

Problem 2 (20 points). Implement the following methods to find the root of a function and test them with a polynomial function of your choice:

(a) Bisection method (`https://en.wikipedia.org/wiki/Bisection_method`).

(b) Secant method (`https://en.wikipedia.org/wiki/Secant_method`).

Problem 3 (20 points). Design and implement a `Money` class for calculations involving dollars and cents where arithmetic has to be accurate to the last cent using the 4/5 rounding rule (.5 of a cent rounds up; anything less than .5 rounds down). Provide addition (`Money + Money`), subtraction (`Money - Money`), multiplication (`Money * int` and `Money * double`), division (`Money / int` and `Money / double`), and equality operators. Represent a monetary amount as a number of cents in a `long`, but input and output as dollars and cents, e.g., $123.45. Do not worry about amounts that don't fit into a `long`. Define corresponding input (>>) and output (<<) operators.

Problem 4 (20 points). Design and implement a `EuropeanOption` class. It should have proper constructor(s) and hold information such as option type (call or put), spot price (of the underlying asset), strike price, interest rate, volatility (of the underlying asset), and time to maturity. Don't accept illegal values. Implement a `getPrice()` function which gives the price of the option using the Black & Scholes formula:

$$
\begin{aligned}
C(S_t, t) &= N(d_1)S_t - N(d_2)Ke^{-r(T-t)} \\
P(S_t, t) &= N(-d_2)Ke^{-r(T-t)} - N(-d_1)S_t \\
d_1 &= \frac{1}{\sigma\sqrt{T-t}}\left[ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \\
d_2 &= d_1 - \sigma\sqrt{T-t}\,,
\end{aligned}
$$

where

- $C(S_t, t)$ is the price of a call option;
- $P(S_t, t)$ is the price of a put option;
- $N()$ is the cumulative distribution function of the standard normal distribution;
- $T - t$ is the time to maturity (expressed in years);
- S_t is the spot price of the underlying asset;
- K is the strike price;
- r is the risk free rate (annual rate, expressed in terms of continuous compounding);
- σ is the volatility of returns of the underlying asset.

Hint: Don't try to implement $N()$ (it is difficult). Do not reinvent the wheel.

Problem 5 (20 points). Add a function to the `EuropeanOption` class that, given an option price, computes the implied volatility of the option using the bisection method. Do the same with the secant method (adding yet another function). Using the provided Options.txt file containing data for European options with different strike values, compute their implied volatilities using both bisection and secant methods. You should output your results to a new .txt file, containing all the content from the original file plus two new columns with the implied volatilities computed by you. Comment on your results.

Hint 1: Use the *mid price* ([bid + ask] / 2) as the option price in your calculations.
Hint 2: The *DaysToMaturity* column of the Options.txt file represents the number of continuous days (not business days) until maturity.

Problem 6 (25 points). Choose 5 or more stocks of your interest and download six months worth of daily stock prices for each stock from Yahoo! Finance (`https://finance.yahoo.com`) as CSV

files. Write a program to pass in the pricing information, then construct a portfolio at the beginning of the 6-month period, consisting of all the stocks you chose. Assume the initial wealth you invest is $\$1,000,000$ and assume you could invest in fractions of a stock, e.g. you could buy 2.46 shares of Netflix (NFLX). **The portfolio weights must be randomly generated.** Calculate the profit if you hold this portfolio for the entire 6-month period without adjusting the portfolio weights. Set different weights to the initial portfolio until you obtain the optimal portfolio. Track the daily values of your optimal portfolio and output them into a CSV file. Use another language (Python, MATLAB, etc) to visualize it. Discuss how you came up with the optimal portfolio.

Note: You should not use maximum return as your objective. Any other definition of "optimal" is encouraged as long as you have your own investment philosophy.