Jake Phelan & Veronica Lekhtman — **CS 3520 Spring 2023 Final Report (HW 6)**

1. Address any comments/requested changes made on your proposal. Describe any major changes to the project after you submitted the proposal. E.g., Did you switch projects? If so, why? (2 points)

We changed the algorithm that we used to price options from Monte Carlo to Black-Scholes. We made this switch because after doing some research, we found that the Black-Scholes algorithm is more numerical and we wanted to use something more formulaic. However, this did not change the end result of our project and turned out to be only a minor change. Additionally, we did not use the Yahoo Finance API to source our stock data, as the API has been discontinued. This also did not end up affecting the end result of our simulation, as we were able to find a dataset with a hefty amount of stock data. After consulting with our instructor, we decided it was sufficient to use static data from the aforementioned dataset.

2. Briefly describe the software you implemented. What does the end result look like compared to the original proposal? (5 points)

We implemented a software which allows users to practice managing an asset portfolio of stocks and options. The users can deposit funds to and withdraw funds from their portfolio's cash account, like in a real stock trading application. This money can then be used to buy and sell both stocks and options, which are volumetrically and monetarily accounted for in the stock portfolio. Users can look at the prices of different stocks before making the decision to purchase them, and can even compare the real price of an option with the fair price for it as calculated by the Black-Scholes algorithm.

3. Describe the high-level design of your source code. For example, what programming paradigms (e.g., functional, object oriented), idioms, and/or design patterns did you use? Describe what each cpp file's purpose is so that we can more easily understand your code. (5 points)

We used some object oriented principles in our project, more specifically we used the model, view, controller design pattern. Trading.cpp and portfolio.cpp act as the implementation of a model that contains all of the necessary methods, where main.cpp functions as a controller that a user can interact with. Trading.cpp includes methods that read in our stock/option datasets and has functionality to run the black scholes algorithm using the data. Portfolio.cpp stores the users portfolio that includes their bought stocks, options, transaction history, and handles all of the methods that involve buying/selling. Main.cpp handles all of the user inputs and uses the methods stored in the model to run the simulation.

4. Provide some instructions for how we should interact with your software in order to see the features you implemented. More detail is usually better. Include screenshots and/or links to short video recordings so that we can see the features you implemented at work even if we have trouble getting your code to compile and run on our computer. For video recordings, a video recorded on your phone is acceptable. (6 points)

Some instructions for interacting with our software can be found in the install.txt file, but we have also produced a YouTube video showing an example interaction with the software.
The video can be found here (in case that doesn't work: https://youtu.be/fdt9Ed1sTv0). We recommend changing the quality setting to 4K or as high as possible in order to read the small font of our terminal.

5. Describe in detail which planned features you implemented, which ones you cut, and which ones you added (if any). Describe these features in a list format, and include the original estimate & priority, any changes you made to that estimate and/or priority during the development process, and how long that feature ended up actually taking you to implement. Include the time you spent on QA in the total time it took you to implement each feature. (8 points)

Original features planned & priority: *estimate changes in bold*

Highest: Option pricing algorithms for fair option pricing, stock portfolio info viewing
Medium: Transaction history, Deposit/Withdraw
Lowest: Individual live-data stock history, front-end software

- Edit and view stock portfolio: 10 hours → **20 hours**
- Deposit/Withdraw currency: 2 hours → **6 hours**
- Check fair option price: 20 hours → **4 hours**

New Features & priority: *new features in bold*

Highest: Option pricing algorithms for fair option pricing, stock portfolio info viewing, Deposit/Withdraw, **buying & selling stocks → 10 hours**
Medium: Transaction history, **stock data as well as option data → 2 hours**
Lowest: **showing all availability stocks & options → 4 hours**

6. Describe your quality assurance process. Did that process change while implementing and testing the software? What QA measures were more or less effective than others? What percentage of the time did you spend on QA? (8 points)

Since our program is dependent on a dataset and complicated formulas, the most effective QA measure we found was to continually run the program with every possible combination of buying, selling, pricing, viewing and manually making sure that the information is correct. We also wrote a few unit tests for our more trivial methods that do not take user input and constructors to make sure that our program did not have any high-level bugs.

7. Describe how you and your partner worked and communicated to complete the assignment. Did you follow your original plan (question 7 from the proposal)? Did your plan change at all, and if so, how? (2 points)

We followed our original plan. We texted often and met up a few times a week to make sure we were both caught up and delegated tasks accordingly. We used GitHub for source control and were able to work on the assignment both together and separately.