CME 211 C++ Exam

Thursday December 5th, 2019

Important note: You are granted **three** hours to complete this exam. For submission instructions, see the submission instruction section at the end of this exam.

1 Short answer questions (27 points total)

For the following section of questions, please record your answers in a `short_answers.md` file. We'd recommend budgeting approximately 30 minutes for this portion; ideal answers will be succinct.

1.1 Declarations and definitions (12 points)

Given the following C++ declarations/initializations:

```
int i, n = 5;
double matrix[4][4], array[3], dt = 0.5;
```

For each statements below (A through D), select one option and justify: (1) the code does not compile; (2) the snippet performs an unsafe operation that will (or is likely to) result in non-deterministic behaviour or crashes; or (3) there are no problems with it. We emphasize that to earn full points you must justify your answer.

(A: 2 points)

```
i = n;
```

(B: 2 points)

```
matrix[4][3] = 2.0;
```

(C: 2 points)

```
array[0] = dt;
```

(D: 2 points)

```
double n = 5;
```

Assuming that the code compiles and `iostream` is included, state what you expect to be printed to console for the following code. Assume `unsigned char` has 8 bit representation. Full points will only be awarded for answers which concisely justify the behavior at the bit-representation level of the data.

(E: 2 points)

```
unsigned char p = 255;
std::cout << (p+1) << std::endl;
```

(F: 2 points) In Natural Language Processing, it's common to construct what's known as a *language model* which can estimate the probability of a word appearing in a sentence. It's possible to estimate the probability of an entire sentence appearing by multiplying the (conditional) probabilities of individual words appearing one after another. Note that the probability of any one word appearing may be very small, and sentences can be arbitrarily long. Can you explain any (numerical) considerations we may wish to consider before naively computing the product of a bunch of small (all non-zero) numbers? To earn full points, you don't have to pose a numerical solution, but you should describe the potential outcome and why it may not be desired.

1.2 Pointers and stack allocated variables (15 points)

(A: 4 points)

State two key differences of memory allocation on the stack and the heap.

(B: 4 points)

Consider the following C++ implementation.

```
#include <iostream>

int* make_triplet(int a, int b, int c){
    int triplet[3];
    triplet[0] = a; triplet[1] = b; triplet[2] = c;
    return triplet;
}

int main(){
    int* triplet = make_triplet(1, 3, 5);
    std::cout << "triplet[1] = " << triplet[1] << std::endl;
    triplet = make_triplet(1, 2, 3);
    std::cout << "triplet[1] = " << triplet[1] << std::endl;
}
```

We compile the code and run the executable, and get the following output:

```
$ ./a.out
triplet[1] = 7827308
triplet[1] = 7827308
```

We intended to print the second element and observe 3 and 2 respectively, but instead we get different values. Explain why the printed values are different from what was intended.

(C: 3 points)

We try to fix the above code, and decide to use keyword `new`:

```
#include <iostream>

int* make_triplet(int a, int b, int c){
    int* triplet = new int[3];
    triplet[0] = a; triplet[1] = b; triplet[2] = c;
    return triplet;
}

int main(){
    int* triplet = make_triplet(1, 3, 5);
    std::cout << "triplet[1] = " << triplet[1] << std::endl;
    triplet = make_triplet(1, 2, 3);
    std::cout << "triplet[1] = " << triplet[1] << std::endl;
}
```

Now the output is

```
$ ./a.out
triplet[1] = 3
triplet[1] = 2
```

as desired. Briefly explain why this corrected the output.

(D: 2 points)

Does the modification introduce a memory leak? Why/why not?

(E: 2 points)

Is there something else that could be done that avoids the use of keyword `new`?

2 Programming (75 points total, inclusive of 15 writeup points)

In this section, your objective is to write a C++ program which can perform basic stock analysis[1]. Use your knowledge of object-oriented programming (OOP) concepts to create a `Stock` class. During grading we will be looking for demonstrated mastery of encapsulation, abstraction, and C++ syntax.

2.1 Overview

Stock Price Time Series The file `msft_close.txt` contains 253 observations of closing prices for Microsoft (MSFT) from Nov 26, 2018 to Nov 26, 2019. Weekends and holidays have been removed, and we will treat each trading day as contiguous for the sake of simplicity in this problem. Here's what the input data looks like:

```
$ head -5 msft_close.txt
106.47
107.14
111.12
110.19
110.89
```

You will modify the provided `main.cpp` file to read all prices from the file into a `std::vector<double>` container.

Daily Return Let p_t be the price at day t. The percentage change in price between day t and $t-1$ is the daily return (1), for which we will use the symbol r_t.

$$r_t = \frac{p_t - p_{t-1}}{p_{t-1}} \tag{1}$$

Returns are important an important concept in finance, because it allows apples to apples comparisons between stocks that have different prices, or strategies with different capital investments.

In this assignment you will write a method to compute the daily returns from stock prices. Note that if you have prices for 100 days, you will only be able to calculate returns for the last 99 days. Do not multiply by 100; use the formula (1) as is.

Mean Daily Return The mean daily return (2) is a simple average over n daily returns which are zero indexed:

$$\overline{r} = \frac{1}{n}\sum_{t=0}^{n-1} r_t \tag{2}$$

You will also write a method to compute the mean return given prices.

[1]Recommended time: 2 hours. Allotment: 30 min to read the instructions and *understand* the problem and solution; 60 min to type out code; 15 min to debug code; and 15 min to justify/write-up `README.md`.

Variance of Returns To measure the risk of a stock, you can compute the variance of its returns series r using formula (3) below.

$$\mathrm{Var}(r) = \frac{1}{n-1}\sum_{t=0}^{n-1}(r_t - \overline{r})^2 \tag{3}$$

You will write a method to compute variance of returns.

2.2 Main

Interface Your program should be called by providing the input file name and stock ticker name as command line arguments. For example, to execute the program on Microsoft data:

```
.\main msft_close.txt MSFT
```

Implementation Modify the provided `main.cpp` to read in a file of stock prices, create a `Stock` object, and then write the stock ticker, mean return, and variance of returns. You should output the results in the order listed here.

Output Format The output should be a text file named `results.txt`, and should follow the following format. You do not need to format the floating point outputs.

```
$ cat results.txt
AAPL
0.0036267
0.0482122
```

Input Assumptions Assume that the inputs your program receives are well formatted, and all prices are non zero.

2.3 Stock Class

Create a `Stock` class to perform the calculations described above. Modify the provided `Stock.hpp` file and create a `Stock.cpp` file. Consider and address ways to avoid repetitive calls to methods, and use OOP concepts like encapsulation in your design.

You may choose the arguments for `dailyReturn`, `meanReturn`, `varReturn` by modifying `Stock.hpp`.

Note that each of the below listed requirements are graded independently, and you can earn points on one section even if you haven't completed the others. You may create any helper methods you need.

Constructor Implement a constructor for a `Stock` object that accepts a `std::vector<double>` container of stock prices and a `std::string` of the stock's identifying ticker.

Daily Returns Create a method `dailyReturn` that calculates daily returns for a stock, and return a `std::vector<double>`.

Mean Return Create a method `meanReturn` that computes the mean return over the available data, and return a `double` .

Variance of Returns Write a method `varReturn` to calculate the variance of the returns, and return a `double`.

2.4 Summary of Coding Requirements (60 points)

1. `main` (10 points)
   - Import stock prices to a `std::vector<double>` container from file.
   - Import the stock ticker from the command line to a `std::string`.
   - Save results to `results.txt` in specified format.
2. `Stock` class (20 points assigned to design; 30 functionality points broken down below)
   - (5 points) A constructor.
   - (10 points) A `dailyReturn` method.
   - (7 points) A `meanReturn` method.
   - (8 points) A `varReturn` method.

2.5 Writeup (15 points)

Design Considerations In the separate markdown file `README.md`, describe and justify your design choices by answering the following questions;

- What data members (attributes, variables) does the `Stock` class have? Are they public or private, and justify your choice.
- What arguments do your `dailyReturn` and `meanReturn` functions accept, and why?
- What considerations did you make to minimize repetitive calls?
- Discuss whether the keyword `new` appears in your program, and why this is appropriate.
- Discuss ONE of the following: (1) an aspect of your program that you are proud of or (2) a possible improvement to your program, or (3) if you did not finish the assignment, the next step needed to fix the program.

Include as part of your `README.md` the command used to compile your program.

Submission Instructions

Within your Github repository, create a directory `exam2`, and add your submission files to it. Your submission should consist of

- `main.cpp`
- `Stock.hpp`
- `Stock.cpp`
- `README.md` file with your writeup and documentation.
- `short_answers.md`

Commit your submission and push it to GitHub. Once when you are satisfied with your submission, tag it "cppexam".

Tags in Github To tag your submission, first make sure that you committed and pushed everything to your GitHub repository. Then, set the tag to the local head by executing

```
git tag cppexam
```

in your local git repository. To push the tag to GitHub, execute

```
git push origin cppexam
```

Using the GitHub GUI to tag a release Note that it's also possible to use the Github web-interface to tag a release: simply navigate to your code on Github, ensure you have selected the `code` tab, then click "release" in the same pane which displays the number of commits and branches. The subsequent page has an icon to "Draft a new release".

Late submissions The time of your submission will be the time of the tagged commit. This time must be less than 3 hours from the time you first accessed the exam from Canvas. We will check out tag "cppexam" from your repository and grade what is there, subject to the submission time being less than or equal to 3 hours from the time-accessed timestamp on Canvas. For exams that roll in after the 3 hours mark, will at first allow a 10 minute grace period, beyond that each additional minute spent on the exam will cost 10 points.