Practical project - Stock Market

420-D02-SU

Introduction to structured programming

AEC Programming, Networks and Security (LEA.5F)

AEC Programming and Web Technologies (LEA.5G)

AEC Video Game Programming (LEA.CU)

**Evaluation weight:**

25% of the final grade

**Submission specifications:**

* **Due date: morning of the final exam**
* Name the program correctly: StockMarket + *name1* + *name2* [ + *name3* ]
* The code of the program and any dependencies
* The compressed file deposited in Dropbox in the folder designated for this purpose

**Description:**

Create a software that simulates a stock market tied to 3 industries. The user can create an account, buy and sell stocks, and witness the changes of the market. Each “Turn” the stock market is affected pseudo randomly by changes to it’s supplies. It is the goal of the user to make the max amount of money in a limited number of turns.

**Remarks:**

* The user should be prompted at the start to create an account.
* A user account contains a name, password, money, and amount in each stock
* The user should be able to output it’s account information, and the value of it’s stocks
* The user should be able to buy and sell desired quantities of stock
* There are a total of 3 stocks that can be purchased
* The user can select “next day” to pass to the next day, which recalculates the value of stocks. (see The Stock Simulation)
* The stock history display screen (See “Stock Display Screen”)
* Input validation and etc
* Make the program scalable for 1->10 stocks (by changing a const value)

**The stock simulation**

Each stock has a hidden array of 5 values, known as “supply values”. These weight values range between 0-3 for each supply value. When a turn happens, a random number (Supply value change) is multiplied by each of these supply value, affecting the stock price.

Example)

In this example, there are two stocks, A and B, you are expected to make 3 stocks.

All 2 stocks have thier hidden supply value array generated (see below)

Stock A has a hidden supply value array {0,2,3,1,0} and a starting price of 20$

Stock B has a hidden supply value array {3,2,2,3,1} and a starting price of 20$

The user creates an account named “MyAccount”, with the password “bob”

The user is asked to re-login to verify credentials.

The user begins with 300$.

User buys 5 stocks of A at 20$ each for a total of 100$.

User selects “next day”

The day begins by creating a random effect to all supplies (-1 || 1) {-1,-1,1,1,-1}

The console outputs the supply effect array.

Stock A multiplies it’s supply value array with that of the day’s supply change to generate its new change in value

{0,2,3,1,0} \* {-1,-1,1,1,-1} = 0 - 2 + 3 + 1 - 0 = 2; //Stock A has increased by two, now 22$

Stock B multiplies it’s supply value array with the same day’s supply

{3,2,2,3,1} \* {-1,-1,1,1,-1} = -3-2+2+3-1 = -1 //Stock B is decreases by 1, 19$

The user checks thier portfolio

Name: MyAccount

Money: 400$

Stock A: 5 Shares at 22$, Worth 120$

Stock B: 0 Shares at 19$, Worth 0$

The user sells 5 stocks of A at 22$ for 110$

The user buys 20 stocks of B at 19$ for 380$

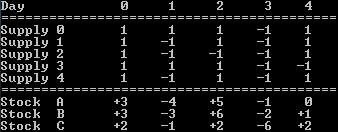
The user selects “Next Day”

The next day occurs, a new random effect to all supplies are generated and displayed.

Stock values are recalculated and balanced

**The stock display screen**

The stock display screen lets you see a history of the stocks performance each day. This screen will help users see the history of stocks and make a decision which stock to buy. The screen needs to maintain formatting with values up to 2 digits.

****

**Picture above is only to show format of stock display screen, it’s values are nonsensical.**

**Evaluation:**

The project will be graded according to the following criteria:

* **Exactitude of the program:**Does the program do what it is supposed to do?
* **Visual appearance:**All text displayed to the screen should be well arranged and written in proper English.
* **Input validation:**The program should properly handle errors that can occur during data input.
* **Structure of the code:**The program should be split up into functions according to the relevant needs.
* **Arrangement and clarity of the code:**The code should be properly indented, with relevant comments where needed, and it should respect programming conventions (variables in camelCase, constants in UPPER\_CASE\_SEPARATED\_BY\_UNDERSCORES). Hard-coded values should be put into constants (magic numbers).

**Teamwork Validation**

This is a team project and must be worked together as a group, and not copied from any external sources.To verify that all members are familiar with the code, after submission each team member will be seperated and questioned about the project. The questions will be along the lines of “Show me the code that does X”, “What will this breakpoint reveal if I do…” or “If I change this code here, what will happen?”. If a member is found being unfamiliar with the code, their total mark will be multiplied by the amount they know. For example, if a member is shown to understand only 30% of the code, and the project received a 90%, their total mark will be 27%.

|  |  |
| --- | --- |
| **Tasks to accomplish** | **Points** |
| Respect submission specifications | 1 |
| Program compiles and executes | 1 |
| Structure of the code (comments, etc) | 2 |
| Account generation & Storage | 1 |
| Output Account info (with stocks) | 2 |
| Buy and sell stocks | 2 |
| Initial generation of Supply Value | 2 |
| Next day flow | 3 |
| Generate Supply Value Change | 2 |
| Calculate new stock values | 4 |
| Display Supply Value Change | 1 |
| Stock Display Screen | 5 |
| Scalability with number of stocks | 2 |
| Input validations | 1 |
| Global evaluation (teamwork, effort) | 2 |
| **Total** (with bonus in parentheses) | 31 |