Project 2 - Financial Literacy

Inspired by: the Freakonomics podcast
Everything You Always Wanted to Know About Money (But Were Afraid to Ask)
and discussion with your peer Ilana Seidl.
0th Milestone is due **Monday 11/15/21** by 11:59 p.m.
1st Milestone is due **Tuesday 11/30/21** by 11:59 p.m.
Complete Project is due **Friday 12/17/21** by 11:59 p.m.

Background

Overarching Background

"If you've spent any time reading up on education or financial news lately, you've probably come across the term *financial literacy*. The goal behind teaching financial literacy is to help people develop a stronger understanding of basic financial concepts—that way, they can handle their money better.

That's a worthy goal, especially when you consider a few stats about how the typical American handles money:(1)

- Nearly four out of every five U.S. workers live paycheck to paycheck.
- Over a quarter never save any money from month to month.
- Almost 75% are in some form of debt, and most assume they always will be.

With those numbers, it's no surprise that leaders in business, education and government want to help spread the benefits of greater financial literacy to as many people as possible.

So, what is financial literacy?

Financial literacy is the possession of skills that allow people to make smart decisions with their money.

And don't be misled by the word *literacy*. Although understanding stats and facts about money is great, no one has truly grasped financial literacy until they can regularly make smart decisions with their money that leads to positive financial outcomes.

When you have this skill set, you're able to understand the major financial issues most people face: emergencies, debts, investments, and beyond. Financially literate people know their way around a

budget, know how to use sinking funds, and can tell you the difference between a 401(k) and 529" (What do you need to know about financial literacy?).

Test your financial literacy

Want to test your financial literacy? Answer these questions and see how well you did here.

1) "Suppose you had $100 in a savings account and the interest rate was 2% per year. After 5 years, how much do you think you would have in the account if you left the money to grow?"
A) More than $102
B) Exactly $102
C) Less than $102
D) Don't know

2) "Imagine that the interest rate on your savings account was 1% per year and inflation was 2% per year. After 1 year, with the money in this account, would you be able to buy..."
A) More than today
B) Exactly the same as today
C) Less than today
D) Don't know

3) "Do you think the following statement is true or false?
Buying a single company stock usually provides a safer return than a stock mutual fund."
A) True
B) False
C) Don't know

If you didn't get all three questions right, you're not alone. This quiz has been used across many countries and demographics, and the majority do not correctly answer all three questions. Notably, "surveys have demonstrated generally lower levels of financial literacy among women, those with low levels of education, and certain minority groups such as African-Americans and Hispanics (Lusardi 11-12). Such barriers to financial literacy are not a matter of intelligence. Instead, traditional gender roles regarding accumulating and managing wealth, socio-economic factors such as income and access to education, cultural attitudes on consumption and financial services, and language barriers often affect the development of financial literacy within those demographics. (Simulations for Financial Literacy)"

If you'd like to read more about how various demographics did on this survey, you can read more here. If you'd like to test your own financial literacy in a game/simulation where you earn minimum wage, click here. (It's very hard.)

Assignment Details

Introduction

In this assignment you're going to be simulating how much money a financially literate versus a non-financially literate person has after 40 years. They both start out with the same amount of money, the same amount of debt, earn the same amount, and buy houses that cost the same amount. The small decisions they make end up accumulating, and by the end of the simulation, you'll see that there's a significant disparity between how much wealth the financially literate and the non-financially literate have.

Financial literacy encompasses a huge amount of knowledge and information. This simulation is *vastly simplifying* the use of money in life in order to make the assignment more manageable and to allow you to see exactly how certain financial decisions have significant outcomes. The three financial literacy questions that you will be exploring through this assignment are as follows:

- Should you put your savings in a bank savings account or in a mutual fund?
- How much of a difference does it make if you pay off your credit card debt now or later?
- Should you wait until you can afford a 20% down payment before you buy a house?

We use a `struct financialIdentity` to represent a person's financial identity. Almost every function that you write will takes in a pointer to such a struct as a parameter. In the simulation, we will have two possible instances of this struct: For simplicity of explanation, we call these `fl` and `nfl`, where `fl` ("financially literate") represents the accounts of a financially literate person and `nfl` ("not financially literate") represents the accounts of a person who is not financially literate. * Both `fl` and `nfl` have a "`savings`" account with a corresponding number that represents the amount in savings they have, a "`checking`" account with money that they use to pay mortgage, "`debt`" from their credit cards (and student loans), a "`mortgage`" for when they take out a mortgage/loan for their house, "`yearsWithDebt`" that will show the number of years each one remains in debt, "`yearsRented`" that will show the number of years that person rented, "`debtPaid`" will show the total amount that the person paid to pay off their debt, and a boolean "`ownsHouse`" that indicates whether the person owns a house or not.

*Coincidentally, most players on the National Football League are **N**ot **F**inancially **L**iterate. 78% of professional football players go bankrupt or are under financial stress within 2 years of retiring, despite having made millions over their careers. You can read more here or listen to a Freakonomics podcast about it called *"Everything You Always Wanted to Know About Money (But Were Afraid to Ask)"*.

Functions

Savings Placement

```
savingsPlacement(struct financialIdentity *person, double
interestRate)
```

This function takes in a `person` struct (which will be either `fl` or `nfl` in the simulation) and a yearly `interestRate`, and it modifies that struct to reflect that person's savings after one year. The only value in the struct that should change is the `savings` account.

Simulation details:

- `fl` and `nfl` both start off with $5000 that they'd like to save.
- `nfl` puts the money in a simple savings account, which (like most saving accounts) has a 1% annual interest. `nfl` doesn't know that since inflation is 2% per year, his money will be worth less and less every year.
- `fl` knows about inflation, so they put their money in a mutual fund. Mutual funds typically have an annual interest rate of 7%.

Paying Off Debt

```
debt(struct financialIdentity *person, double interestRate,
double addlPay)
```

Write a function that applies monthly payments and updates the amount of remaining `debt` a `person` has and the amount of money they have left in their `checking` and `savings` accounts **after one year**. `debt` models credit card and student loan debt, which for simplicity we've combined in the single entry called `debt`.

To apply payments, update the `debt`, `debtPaid`, and any other relevant entries **per month and create a loop that runs it for 12 months**. *Do not try and calculate the total for the year without using the loop, because you will get a different amount.* (Note: The loop should stop on the *month* that the debt is no longer over $0.) The `yearsWithDebt` entry should be updated for the `person` **each year**. If `fl` or `nfl` doesn't have enough money in their `checking` account to make a debt payment, they use their `savings` account to pay the difference.

The interest on the debt that hasn't been paid off is compounded annually. This is calculated each year by multiplying the remaining `debt` by 1 + `interestRate`.

Simulation details:

- Initially, `fl` and `nfl` both have $52,500 of debt on their credit cards and in student loans, which for simplicity we've combined under the single entry called `debt`.
- Each month, the minimum amount that the credit card company requires to be paid off is 3%. `nfl` pays off just the minimum 3% each month plus an additional dollar (the additional dollar is important because without it, the function would continue taking 3% of the remaining debt, and consequently would approach 0 but never reach it). `fl` pays off the minimum 3% plus an additional $100 of it, knowing that paying the debt off more quickly will reduce the amount that they ultimately end up paying.
- The `interestRate` is 20%. So at the end of the year, `debt` is multiplied by 1.2.

Different Down Payments on a House

Simulation details:
`fl` and `nfl` decide that they'd like to buy houses. The difference is that they decide to buy them at different times in their lives. `nfl` buys the house as soon as they're able to afford a down payment of 5% on the house (so as soon as their savings account is over 5% of the total cost of the house). `fl` buys the house as soon as they're able to afford a down payment of 20% on the house (so as soon as their savings account is over 20% of the total cost of the house). Both are buying a house that costs $210,000, and the interest rate for both is 4.5%. What `nfl` doesn't know is that when you pay a down payment less than 20% on any house, an extra interest rate called "private mortgage insurance" (PMI) is added onto the cost. In that case a 0.5% (assuming `nfl` has a great credit score) interest rate is added onto the amount that `nfl` has to pay each year.

Before each one moves into a house, they're living in apartments that cost them $950 a month in rent (the average affordable amount in the US).

```
rent(struct financialIdentity *person, double rentAmount)
```

Write a function that updates the amount of money a `person` will have left in their `checking` and `savings` accounts **after one year**, if they're in the rented apartment. `rentAmount` is the **monthly** rent amount. The `yearsRented` entry should be updated for the `person` **each year**. If `fl` or `nfl` doesn't have enough money in their `checking` account to pay for rent, they use their `savings` account to pay the difference.

Simulation details:

- For both `fl` and `nfl` the cost per month to rent is $950.

```
house(struct financialIdentity *person, double
mortgageAmount, double interestRate, int mortgageTerm)
```

Write a function that updates the amount of money a person will have left in their `checking` and `savings` accounts **after one year**, if they live in a house and thus pay mortage payments on that house. Note, `mortgageAmount` is the house price minus the down payment, `interestRate` is the yearly interest rate, and `mortgageTerm` is the number of years that the mortgage has to be paid off by.

To apply mortgage payments, update the `mortgage` and any other relevant entries **per month and create a loop that runs it for 12 months.** *Do not try and calculate the total for the year without using the loop, because you will get a different amount.* (Note: The loop should stop on the *month* that the mortgage is no longer over $0.) If `fl` or `nfl` doesn't have enough money in their `checking` account to make a mortgage payment, they use their `savings` account to pay the difference. Here are formulas to calculate the **monthly mortgage payment** (which is also explained here):

- $\#\ payments\colon N\ =\ mortgageTerm\ times\ 12\ (so\ if\ mortgageTerm = 30,$ $this\ is\ 360 =\ 30\ years\ times\ 12\ monthly\ payments\ per\ year)$
- $monthly\ interest\colon i\ =\ (yearly\ interestRate\ divided\ by\ 12)$
- $discount\ factor\colon D\ = ((1+i)^N - 1)\ /\ (i(1+i)^N)$
- $monthly\ payment\colon P\ =\ mortgageAmount\ /\ D$
- This monthly payment amount is the same for the entire life of the mortgage.
- Don't forget to account for the monthly interest that is added to the mortgage throughout its life.

Simulation details:

- To calculate the monthly mortgage payment, you can assume that both `fl` and `nfl` will pay off their mortgages in full in `mortgageTerm` = 30 years and that their payments will be the same every month (according to the formula above).
- The interest on the mortgage for `fl` is 4.5%. The interest on the mortgage for `nfl` is 5% (that is 4.5% for the mortgage and 0.5% of PMI).
- Don't forget to account for the yearly interest that is added to the mortgage throughout its life.

(You can read more about buying a house with different down payments here.)

40-Year Simulation

```
simulate(struct financialIdentity *person, double
yearlySalary, bool financiallyLiterate):
```

For one year, these small choices that `fl` and `nfl` make may not have a huge effect on their savings and checking accounts, but over forty years you can see a much larger effect. Now you're going to be using all of the previous functions as helper functions to simulate how much `fl` and `nfl` will have after 40 years.

`fl` and `nfl` both earn $69,000 a year (the median income of a middle-class person). They receive the money at the start of each year. They put 20% into `savings`, and 30% into `checking`. In writing the function, you can pretend that this 50% is all that they earn, since the other 50% is theoretically being used up elsewhere. (15% is spent on food. 15% is spent on transportation. 10% is spent on miscellaneous. 10% is collected for taxes. So, the amount that ends up going into checking and saving combined at the end of each year is actually $34,500.)

Details:

- `simulate(struct financialIdentity *person, double yearlySalary, bool financiallyLiterate)` should return an array with 41 elements. Each element represents how much "wealth" the inputted person had that year. (These numbers will later be plotted in a graph.)
- In each year, "wealth" should be a single number that represents the total money the person has in their accounts (checkings + savings – debt - mortgage) that year, and it should be rounded to an int. The initial wealth (- $47500 = initial savings ($5,000) - initial debt ($52,500)) should be the first element in the array.
- Each year, the person should put 20% of their `yearlySalary` ($69,000 for `fl` and `nfl`) into savings, and 30% of their `yearlySalary` into checking.
- Run `savingsPlacement( )` and `debt( )` for each simulated year.
- Before the person has enough money in their savings account to pay the 5% or 20% down payment, the person should pay `rent()` each year. Once they have enough for the down payment, `mortgage` should be updated to be the price of the house minus the down payment, and then `house()` should be run each year until the mortgage has been paid off.
- You should return the array at the end, not for each iteration.

Technical details

You will not have to return anything for any of the functions except `simulate()`. You can pass numeric types to functions by value, **but structs should be passed by pointer**. This is very important! If you don't pass pointers, any changes you make to the struct within a function won't affect that struct outside of it.

You might want to define `printWealth()` and `printPerson()` functions for debugging.

Input and output

You should develop and test your simulation using the sample numbers in this document.

The wealth arrays are best analyzed visually, so the wealth values need to be output to a file, one number per line. There should be one file for `fl` and one file for `nfl`. Your program must take in the name of your output files as command line argument. Assuming that we've compiled your source into an executable named `FinancialSimulation` and that we want to write to output files `output_fl.txt` and `output_nfl.txt`, we will expect to be able to run your program on the terminal as follows:

```
$ ./FinancialSimulation output_fl.txt output_nfl.txt
```

Written Response

Answer the following questions in writing:

1. What did you learn from this assignment that you didn't know before?
2. How much more debt did `nfl` end up paying compared to `fl`?
3. How many more years was `nfl` in debt compared to `fl`?
4. How much more money does `fl` have at the end of 40 years compared to `nfl`?
5. You don't need to test each function to answer this question, but which decision do you think had the greatest effect on the disparity between `fl` and `nfl`?
6. Come up with three life financial decisions (that were not mentioned in this assignment) that a non-financially literate person and a financially literate person might make differently. Specify which person would make which decision.
7. Plot the wealth of `fl` and `nfl` using your favorite graphing software (e.g. Excel).

Potential Additional Features for Extra Credit

Extra credit should only be pursued if the main program runs. Here are a few ideas for extra credit opportunities, but feel free to come up with others:

You might want to explore scenarios with different values for `yearlySalary`, mortgage `interest rate`, `rentAmount`, house price, initial debt, and additional debt payments (`addlPay` for the `debt` function), for example.

You may want to expand your program to read in an input file which provides these values for your simulation. The name for the input file could be an additional command line argument.

You could plot the results for different scenarios and compare the scenarios in a written document.

Milestones

For the 0-th milestone, you will submit the answers to the three financial literacy test questions (on page 2 of this document). And you will submit a header file `FinancialSimulation.h` which contains (1) the `struct financialIdentity` with all its variables as specified in this document, and (2) the headers (stubs) for all the functions specified as specified in this document: `savingsPlacement`, `debt`, `rent`, `house`, `simulate`.

For the 1st milestone, your program should import your `FinancialSimulation.h` (which means your `financialIdentity` struct has been defined with all its variables and all function headers (stubs) have been specified). And you should at minimum have implemented the `savingsPlacement` and `rent` functions, and at least a draft (or first attempt) of the `debt` function.

Submit

For your final submission, submit the following in a compressed zip archive:

- Main code file(s). Code should be clean, well-commented, have meaningful variable and function names, not have commented-out code. Code should be easily readable by a human. Main code should be broken into functions that perform logical parts of the job. Most importantly, the code should compile and run.
- The generated output files.
- A README.md file containing explanation of your program, known extensions and limitations, and instructions how to run the program.
- A .pdf file containing answers to the written response part including a plot of wealth for two people (`fl` and `nfl`) under the conditions described in the assignment.

Tentative Rubric

- 0-th milestone 10 points
- 1st milestone 10 points
- Final submission total 80 points
  - `savingsPlacement()` 5 points
  - `debt()` 10 points
  - `rent()` 5 points
  - `house()` 10 points
  - `simulate()` 15 points
  - `main()` 5 points
  - Code quality / documentation (including README) 10 points
  - Output (including command line argument) 10 points
  - Written response 10 points

- Additional features, creativity, and innovation: up to 20 points