Name: Olivia Nye

Date: 04/09/2019

Class: CMPS 1600 Spring 2019

Project 1: Financial Literacy

**Program Overview**

This project, coded in C, demonstrates the importance of financial literacy in long term wealth. This project runs 40-year wealth simulations, using core functions debt, house, rent, and savingsPlacement, as well as other functions, for persons created in a struct. For generalizability, the main function has been equipped to display a user-friendly interface that allows for the execution of simulations for a variety of persons (2 hard-coded persons, 4 persons that can be created from existing input text files, and ability to create persons from user input). All simulations output wealth arrays to text files. This program has passed thorough manual testing in separate files that were excluded from the submission, in accordance with the instruction to exclude commented out code.

**Instructions for Running the Program:**

To run the program, access an online or offline C IDE (repl.it is preferred. Please avoid using CLion if possible.) If using an offline IDE, ensure that you have a functional C compiler installed. Open the main.c file in your IDE and ensure that the four .txt input files submitted with the project (“flPovertyPerson.txt”, “nflPovertyPerson.txt”, “flRichPerson.txt”, “nflRichPerson.txt” are in your workspace, or upload them to the project if using an online compiler. Run the main.c file and follow the menu options that appear. Run as many simulations as you like using the menu options. When creating your own person with menu option 7, read the instructions carefully to make sure you enter your information in the proper format. Output .txt files will appear/update after you quit (as explained in the menu, quitting is done by entering menu option “0” when presented with the menu of actions).

**Extra Credit:**

For my extra credit functionality, I decided to implement my creativity to make my main function extremely user friendly. I made the main function display conversational text elements and I developed a menu of simulations that the users can run by entering menu options. The most considerable extra credit addition I made was option 7, which tests the user for their financial literacy using the questions from the project assignment, takes user input for person struct values, runs the simulation for the user, outputs their wealth array. I believe that this additional feature adds substance to the project because it makes exploration of the interaction between wealth and financial literacy accessible and easily understandable to users, even to those with no/minimal competency in the worlds of finance and computer science. Additionally, I found my decision to use 2 persons with poverty-level salaries and 2 persons with salaries comparable to that of the top 1% of the US population (I chose to do some extra research on this and I obtained these values from US census records) for my additional persons from input files to be quite innovative. These functionalities offer an interesting look into the relationship between socioeconomic status and potential wealth, and how that relationship can be moderated by financial literacy.

**Function Descriptions (in order of appearance in main.c -- not the order in which functions are executed):**

* Void savingsPlacement function takes in pointer to person struct and an interest rate. Function calculates the amount of money left in the person's savings after collecting interest
* Void debt function makes monthly payments (the minimum 3% payment plus an additional amount specified by the person) from savings account to pay off credit card debt while updating the values of person's debt, debtPaid, savings, and years with date. Function accounts for annually compounded interest
* Void rent function takes in pointer to person struct and a rent amount. Function pays rent for 12 months and removes the money used to pay rent from the person's checking account. Function also keeps track of how many years the person has rented
* Double powerOperation function performs the power mathematical operation. The argument to the first parameter is the number you want to multiply by itself. The second argument to the parameter is the number of times you want to multiply the first parameter by itself.
* Void housingPayments function takes in pointer to struct person and determines how much the person should pay towards their loan every month in order to pay off their loan in exactly 30 years.
* Void makeDownPayment function takes in pointer to struct person and removes the person’s downPayment amount from their checking account and sets the value of the person's loan. This function is called by the contemplatePurchase function (see below) when the person is ready to buy a house.
* Void contemplatePurchase function takes in pointer to struct person and decides whether or not a person is ready to buy a house based on their financial literacy, the price of the house they want to buy, and the amount of money in their checking account.
* Void house function pays a years worth of mortgage payments for the person (stops when loan is paid off if that happens in the middle of the year). Each month, function removes the payment amount from the person's checking, subtracts that amount from their loan total, adds that amount to their loanPaid, and adds 1/12 of the yearly added interest to their loan total.
* Void setWealth function takes in a pointer to struct person and calculates their current wealth by finding the sum of their savings and checking and subtracting their debt and their loan.
* Int simulator function executes the 40 year wealth simulation. Function calculates initial wealth for the person and then simulates 40 years of financial actions by calling the other functions defined in this project. The person's wealth is calculated and added to an array of wealth values at the end of each year.
* Void outputWealth function takes in a pointer to struct person and an output text file and prints the wealth array (one number per line) to the text file.
* Void newPersonInput function takes in a pointer to struct person and an input text file, reads the file to obtain 7 values and assigns those values to their respective struct variables for that person.
* Void printMenu function is used in main to repeatedly print the menu of actions for the user to choose from
* Int main function displays a user-friendly interface that allows for the execution of simulations for a variety of persons (hard-coded fl and nfl, 4 persons that can be created from existing input text files (a financially literate person living in poverty, a not financially literate person living in poverty, a financially literate wealthy person, and a not financially literate wealthy person), and persons created from user input). All simulations output wealth arrays to text files. Option 7 tests user for their financial literacy, allows for users to create their own person struct by providing user input, runs the simulation.

**Limitations:**

As it stands, this program is fully functional. Despite this, the wealth outputs are largely overestimates. This program does not account for inflation, insurance payments, rises and falls of the economy, or a variety of other important financial decisions like budgeting, getting married, and having children.

Additionally, it appears that this program will not work on cLion due to issues with taking input from the command line and files. I believe this problem only occurs because of errors with the compiler on my machine, and I was instructed by TAs not to be concerned about it because my program is fully functional with Repl.it. However, occasional Repl.it bugs prevent the output files from appearing, but this bug consistently fixes itself when the page is reloaded and the program is rerun, without having to make any changes to the code.

**Sample Input File Text:**

“

30000.0

3000.0

1

5000.0

1500

2000000

300000.0

“

These numbers represent the values for the following person struct variables (in order): double savings, double debt, double checking, int rent, int housePrice, double yearlySalary.