Functions

Banking App V1.0

Workshop 5 (worth 3% of your final grade)
URL: https://github.com/Seneca-144100/IPC-WS6

In this workshop, you are to write an application that handles banking services for Bank ABC. Your application allows banking staff to:

- Deposit cash for customers
- Withdraw cash for customers
- Add monthly interest earnings to all accounts
- Apply service charges to all accounts
- Display account details

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities to:

- Define and implement functions
- Call functions with several input parameters and receive data back
- Understand reusability of logic through modularity

SUBMISSION POLICY

Your workshops are divided into two sections; in_lab and at_home.

The "in_lab" section is to be completed **during your assigned lab section**. It is to be completed and submitted by the end of the workshop. If you do not attend the workshop, you can submit the "in_lab" section along with your "at_home" section (a 20% late deduction will be assessed). The "at_home" portion of the lab is **due the day before your next scheduled workshop**

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible for regularly backing up your work.

IN-LAB: ITEM CLASS (70%)

Download or clone workshop 5 from https://github.com/Seneca-144100/IPC-WS6

Write this section of your code in banking.c provided with the visual studio template project inside in_lab folder.

In this workshop segment, you will implement deposit, interest earnings and account display functionality for savings accounts in the Banking App and will use C functions to modularize your solution.

Overview

The banking.c template file has already implemented the following:

- Display an option list as following inside a do-while loop construct:
  1.) Deposit
  2.) Withdraw
  3.) Apply monthly interest earnings to all accounts
  4.) Apply service charges to all accounts
  5.) Account Summary
  0.) Log out
- Capture a user input for the above options
- Ability to iterate over your inputs and then branch in to relevant switch cases (6 cases including default). The switch-cases are furnished with comments as to what their functionality should be.
- Display an error messages if your input is not a valid option
- Exit the program on option 0.want
- The template program has 5 savings accounts already defined and initialized with the following information using a **`struct account`** array named **acct.** **`struct account`** has two member variables named **`int acctNumber`** and **`double balance`**.

accNumber (account number)	**Balance**
11111111	123.45
22222222	12365.50
33333333	0
44444444	1475
55555555	25000.65

You are required to complete the following functionality.

Implement deposit functionality in Case 1

- Prompt the user to input the **accNumber** number
- Search through all accounts (**`acct`** array) to find whether the account exists.
- If the account is found, then do the following:
    - Prompt for an amount to deposit
    - Define and implement a C function called **balanceUp** above your **main** function with two input parameters "**`double balance`** and **`double amount`**". This function adds the values of both input parameters *only if* **`amount`** *is positive-valued*; otherwise, this function does not alter the balance. This function returns the resulting balance as a **`double`**.
    - Call **balanceUp** with the account balance and the amount to deposit as arguments inside case 1 in the **main** function. Assign the returned result as new balance back to the relevant account.
    - Display a message showing the new balance for that account.
- If the account is not found, display an error message saying the account does not exist.
- Please refer to the sample output below for exact message contents and formatting

Implement interest earnings functionality in Case 3

- Iterate through all accounts. Find the relevant interest rate depending on the balance of each account as per the following table

balance	Interest rate
`balance = <500`	`0.99`
`balance >500 && balance =< 1500`	`1.66`
`balance >1500`	`2.49`

- Define and implement another C function **interestCalc** above your **main** function with two input parameters "**`double balance`** and **`double rate`**" to find the monthly interest portion (**`balance * (rate/100)`**) and return this interest portion as a **`double`**.
- Call **interestCalc** with the **account balance** and the **interest rate** as arguments inside case 3 in the **main** function. Assign the returned result in to a temporary variable **double calcInterest**
- Call your **balanceUp** function with the account balance and **calcInterest** as arguments inside case 3 in the **main** function to add the interest earnings to the bank balance. Assign the returned result as new balance back to the relevant account
- Display the following for each account. Use `%8d %11.2lf %21.2lf` to format display

```
 Account# New Balance Interest Earnings (M)
-------- ----------- --------------------
11111111      583.65                 9.53
22222222    12673.40               307.90
.. More
```

- Perform the above for all accounts

Implement account display functionality in Case 5

- Iterate over all accounts. And display account information as per following format

```
Account# Balance
-------- ----------
11111111     574.12
22222222   12365.50
33333333       0.00
44444444    1495.00
55555555   25000.65
```

- Use `8d %10.2lf` to format display the above

Program completion

Your program is complete, if your output matches the following output. Red numbers show the user's input.

```
***** Welcome to Savings Account Banking *****

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 5

-- Account information --

Account# Balance
-------- ----------
11111111     123.45
22222222   12365.50
33333333       0.00
44444444    1475.00
55555555   25000.65

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 1

-- Make a deposit --

Enter account number: 67676767
ERROR: Account number does not exist.

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
```

```
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 1

-- Make a deposit --

Enter account number: 44444444
Enter amount to deposit (CAD): 20
Current balance is : 1495.00

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 1

-- Make a deposit --

Enter account number: 11111111
Enter amount to deposit (CAD): 450.67
Current balance is : 574.12

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 5

-- Account information --

Account# Balance
-------- ----------
11111111     574.12
22222222   12365.50
33333333       0.00
44444444    1495.00
55555555   25000.65

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 3

-- Add monthly interest earnings to all accounts --

Account# New Balance Interest Earnings (M)
```

```
-------- ----------- ---------------------
11111111      583.65                  9.53
22222222    12673.40                307.90
33333333        0.00                  0.00
44444444     1519.82                 24.82
55555555    25623.17                622.52

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 5

-- Account information --

Account# Balance
-------- ----------
11111111     583.65
22222222   12673.40
33333333       0.00
44444444    1519.82
55555555   25623.17

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 0
```

For submission instructions, see the SUBMISSION section below.

IN_LAB SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above or any information needed.

If not on matrix already, upload your **banking.c** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

~profname.proflastname/submit ipc_w6_in_lab <ENTER>

and follow the instructions.

AT_HOME: TITLE (20%)

Implement the following cases to the same **banking.c** file which you already completed the in-lab exercise. The **sample** input/output comments at the bottom of **this file** should now be replaced with the "Program Completion" input/output shown below.

Implement withdrawal functionality in Case 2

- Prompt the user to input the **accNumber** number
- Search through all accounts (`acct` array) to find whether the account exists.
- If the account is found, then do the following:
    - Prompt for an amount to withdraw
    - Define and implement a C function called **balanceDown** above your **main** function with two input parameters "`double balance` and `double amount`". This function subtracts **amount** from **balance** *only if* `amount` *is positive-valued*; otherwise, this function does not alter the balance. This function returns the resulting balance as a `double`.
    - Call **balanceDown** with the **account balance** and the **amount to withdraw** as arguments inside case 2 in the **main** function. Assign the returned result as new balance of the relevant account.
    - If the withdrawal amount is higher than the balance **balanceDown will return a negative number**. In this case, display an error message saying that the withdrawal is unsuccessful along with the maximum amount that can be withdrawn (= balance)
    - Display a message showing the new balance for that account.
- If the account is not found, display an error message saying the account does not exist.
- Please refer to the sample output below for exact message contents and formatting

Implement service charge functionality in Case 4

- Iterate through all accounts. Find the relevant service charge depending on the balance of each account as per the following table.

balance	Service Charge
balance = <1500	7.50
balance >1500	2.50

- Then call **balanceDown** again with the **account balance** and **service charge** as arguments inside case 4 in **main** function to deduct the service charge from the bank balance. Assign the returned result as new balance back to the relevant account
- Display the following for each account. Use `%8d %11.2lf %19.2lf` to format display

```
Account# New Balance Service charges (M)
-------- ----------- -------------------
11111111      326.15                7.50
22222222    12670.90                2.50
.. More
```

- Perform the above for all accounts

Program completion

Your program is complete if your output matches the following output. Red numbers show the user's input.

```
***** Welcome to Savings Account Banking *****

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 5

-- Account information --

Account# Balance
-------- ----------
11111111     123.45
22222222   12365.50
33333333       0.00
44444444    1475.00
55555555   25000.65

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 1

-- Make a deposit --

Enter account number: 44444444
Enter amount to deposit (CAD): 20
Current balance is : 1495.00

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 1

-- Make a deposit --

Enter account number: 11111111
```

```
Enter amount to deposit (CAD): 450.67
Current balance is : 574.12

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 5

-- Account information --

Account# Balance
-------- ----------
11111111     574.12
22222222   12365.50
33333333       0.00
44444444    1495.00
55555555   25000.65

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 3

-- Add monthly interest earnings to all accounts --

Account# New Balance Interest Earnings (M)
-------- ----------- --------------------
11111111      583.65                 9.53
22222222    12673.40               307.90
33333333        0.00                 0.00
44444444     1519.82                24.82
55555555    25623.17               622.52

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 5

-- Account information --

Account# Balance
-------- ----------
11111111     583.65
22222222   12673.40
33333333       0.00
44444444    1519.82
```

```
55555555   25623.17

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 2

-- Withdraw funds --

Enter account number: 67676767
ERROR: Account number does not exist.

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 2

-- Withdraw funds --

Enter account number: 11111111
Enter amount to withdraw (CAD): 250
Current balance is : 333.65

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 2

-- Withdraw funds --

Enter account number: 33333333
Enter amount to withdraw (CAD): 4500.56
Withdrawal failed. You only have : 0.00 in your account

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 2

-- Withdraw funds --

Enter account number: 55555555
```

```
Enter amount to withdraw (CAD): 6200.40
Current balance is : 19422.77

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 5

-- Account information --

Account# Balance
-------- ----------
11111111     333.65
22222222   12673.40
33333333       0.00
44444444    1519.82
55555555   19422.77

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 4

-- Apply service charges to all accounts --

Account# New Balance Service charges (M)
-------- ----------- ------------------
11111111      326.15                7.50
22222222    12670.90                2.50
33333333       -7.50                7.50
44444444     1517.32                2.50
55555555    19420.27                2.50

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 5

-- Account information --

Account# Balance
-------- ----------
11111111     326.15
22222222   12670.90
33333333      -7.50
44444444    1517.32
```

```
55555555   19420.27

1.) Deposit
2.) Withdraw
3.) Apply monthly interest earnings to all accounts
4.) Apply service charges to all accounts
5.) Account Summary
0.) Log out

Please enter an option to continue: 0
```

AT-HOME REFLECTION (10%)

Please provide brief answers to the following questions in a text file named **reflect.txt.**

1) What is a function and briefly discuss the need for functions in any language?
2) Comment on the modularity and reusability of **balanceUp** and **balanceDown** functions. (Hint: Each function was called twice for different purposes each time)
3) How do you send and receive data back to and from functions.

AT_HOME SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload your **banking.c and reflect.txt** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

~profname.proflastname/submit ipc_w6_at_home <ENTER>

and follow the instructions.