CSE102 – Computer Programming (Spring 2022)

Homework #9

Handed out: May 15, 2022.

Due: 11:55pm May 25, 2022.

Hand-in Policy: Via Teams. No late submissions will be accepted. Files named main.c and makefile for this homework and compress it into a StudentNumber_Name_Surname.zip file. For your questions, you can write under the assignment post in the Teams application.
Collaboration Policy: No collaboration is permitted.
Grading: This homework will be graded on the scale of 100.

Homework Description: In this homework, you will implement a banking system. This homework will be implemented by using Structs, Unions, Recursive functions and File operations in C.

You need 2 unions and 1 struct type. These are;

```c
union Person
{
    char name[50];
    char address[50];
    int phone;
};
union Loan
{
    float amount;
    float interestRate;
    int period;
};
struct BankAccount
{
    union Person Customer;
    union Loan Loans[3];
};
```

The first union structure is Person. This union has to keep personal information. This value may change, in the image, it is for example purposes only. You can add extra information. The second union is credit. Here, it should keep the amount of the loan, the interest rate and the number of periods. The struct is BankAccount. Inside the struct, you have to keep Person and Loan of type union. You will open bank accounts within the system and assign loans to customers by calculating loans. Each customer can only have 3 loans. Banking system keeps maximum 50 customer in same time. Each fields are will be get from user input with using command line.

Function prototypes are :

listCustomers()

addCustomer ()

newLoan ()

calculateLoan(float amount, int period, float interestRate);

getReport();

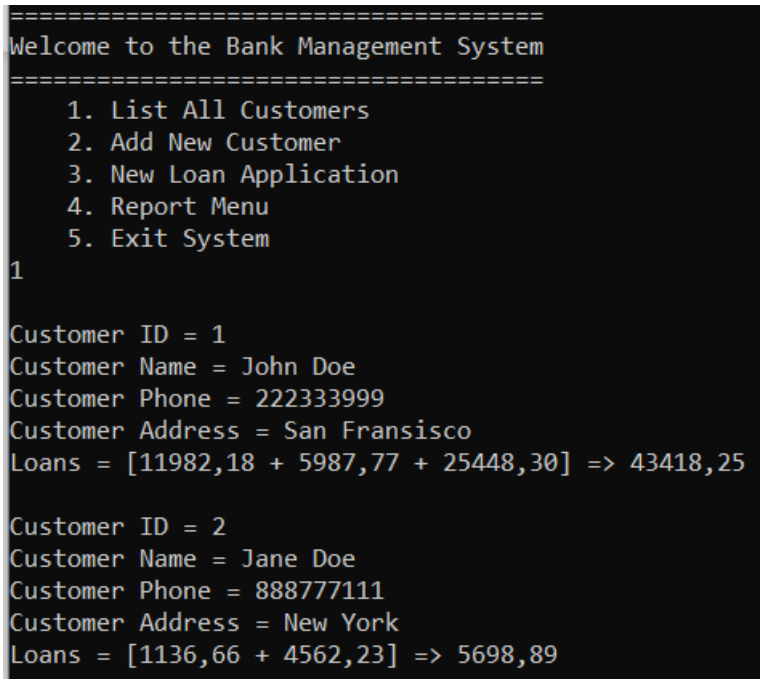
You have to use a recursive function when calculating the loan. You are not allowed to use the pow() function. The formula for calculating the loan is below.

$$\boldsymbol{Loan\ Formula} = \boldsymbol{Amount} * (\mathbf{1} + \boldsymbol{interestRate})^{\boldsymbol{period}}$$

In the report section, you have to print out 2 reports. These are the Customer list and Loan detail. You must keep all customers in the customers.txt file. When requesting this report, you must read customers.txt file and print the customers to the screen. When printing a loan detail of a customer, you need to print the total value of that loan and each period one by one as in the example. Not all states are shown in the expected outputs, but you have to check all states and incorrect entries for each step. You can specify the function types according to your needs. Unless the program exit option is selected, it has to return to the menu after each operation. Remember, the list in option 1 has to print the customers in the struct to the screen. The report menu in the 4th option has to print the customers and loans it reads from the file on the screen.

Expected Outputs

```
Total Credit Value = 11956.1807
1. Month Installment = 996.3484
2. Month Installment = 996.3484
3. Month Installment = 996.3484
4. Month Installment = 996.3484
5. Month Installment = 996.3484
6. Month Installment = 996.3484
7. Month Installment = 996.3484
8. Month Installment = 996.3484
9. Month Installment = 996.3484
10. Month Installment = 996.3484
11. Month Installment = 996.3484
12. Month Installment = 996.3484
```



General Rules:

1. Obey the style guidelines.
2. Do not change the provided function prototypes (you will not get any credits).
3. The program must be developed on Ubuntu using GCC compiler (version provided in class), compilation problems due to the use of another OS or compiler is your responsibility (you will not get any credits).
4. Your program should work as expected. Do not expect partial credit if your code works only in some cases but not in all cases as expected.
5. Hand in your work using the appropriate class Teams assignment site.
6. No late submissions will be accepted.
7. Pack this directory into a zip file named 20180000001_X_Z.zip
8. When unpacked as above in Ubuntu (version provided in class) it should allow executing the following commands in a shell:
   - "$make clean" removes everything except makefile, source code (.c) and other resource files (if any) – all compiling results and intermediate files should be removed (except results.txt).
   - "$make compile" should compile the code.
   - "$make run" should run the code along with any parameters needed.