CSE 2025 – PROJECT1

Due Date: 14.05.2023 23:59

Project Description

In this project, a loan withdrawal system will be simulated. In this loan system, customers will be able to choose different types of loan. A customer can choose a loan from more than one loan type. Installments are created in the payment plan for each selected loan. In the system, customer information and loan information withdrawn by each customer are kept using the linked list data structure. Installments are created for each loan withdrawn and the installments are also kept using linked list data structure. The mentioned linked lists are created using structs and pointers. In the system, the customer information, the loan information withdrawn by the customers and the payment information of the installments of the loans are read from the input files. The information read from the input files is kept as a linked list using structs and pointers. Information about the structs to be used is given below.

Structs:

1. customer:

This struct is used for keeping customer related information.

```
struct customer {
    char name[20];
    char surname[30];
    int customerid;
    char customertype[20];
    struct customer *nextcust;
    double totaldebt;
    struct loan *loanptr;
}
```

Customer name and surname are stored using **name** and **surname** arrays, respectively. Each customer is given an id that is stored in **customerid** data field and customer id starts from 1. Type of the customer (i.e, bireysel, kurumsal_kucuk, bireysel_vip etc.) is stored in **customertype** array. Total debt of the customer is stored in **totaldebt** data field. Total debt is initially equal to zero but it is calculated and updated for each customer later. Loans of the customers are kept as a linked list using **loanptr** pointer. Customers are kept as a linked list using **nextcust** pointer to point to the next customer in the system.

2. loan:

This struct is used for keeping loan related information.

```
struct loan{
        char loanid[30];
        char type[30];
        float totalamount;
        int totalinstallmentnum;
        char processdate[11];
        struct loan *nextloan;
        struct installment *insptr;
}
```

Each loan has a specific id (**loanid**), type (**type**), total amount (**totalamount**), number of installments that are going to be created for this loan (**totalinstallmentnum**) and date at which this loan has been withdrawn (**processdate**) by the customer. Loan pointer of a customer (**loanptr**) points to the linked list of loans withdrawn by that customer. Each element in the linked list is type of **struct** loan and each loan in the linked list points to the next loan withdrawn by the customer. Therefore, **nextloan** pointer is used for pointing to the next loan in the list. Installments created for the loan are kept as a linked list using a pointer (**insptr**) type of struct installment. Loans are inserted into the list in a sorted way so that they are sorted from oldest to newest.

3. installment:

This struct is used for keeping installment information of each loan.

```
struct installment {
        char insid[30];
        short ispaid;
        char installmentdate[11];
        float amount;
        struct installment *nextins;
}
```

Each installment has a specific id (**insid**), a due date (**installmentdate**) whose format is dd/mm/yyyy (day/, month/ and year, respectively) and amount that is going to be paid (**amount**). Whether the installment is paid or not is also recorded using **ispaid** data field. Each installments belonging to the same loan are kept as a linked list and each installment points to the next installment in the list with **nextins** pointer.

Simulation:

When the simulation is run, a menu showing the options is shown to the user. When the user selects an option corresponding task is performed by the simulation and **the menu is shown again until the user enters 0**. Each time an option is selected a corresponding function will be called. The function names that will be called for each option are given in the sections where the options are explained.

```
################################################################
1. read customers.
2. print customers.
3. read loans.
4. print loans.
5. create installments.
6. print installments.
7. read payments.
8. find unpaid installments.
9. delete completely paid installments.
please select your option :
1
################################################################
```

1.option 1:

When the option 1 is selected by the user, **customers.txt** file is read and a linked list of customers is created. To accomplish this task, a function called as **readCustomers** is called in the main method. Id of the first read customer is 1 and id of the customer is incremented by one each time a new customer is read and placed into the linked list. ***totaldebt*** of each customer is 0 initially. Head pointer of the linked list is called as **customers** and given in main function of the sample code. Each linked list members are connected to each other using the ***nextcust*** pointer. There can be more than one customers having either the same name or surname but there will not be two customers having both same name and surname.

customers.txt file:

This file contains the **name**, **surname** and **type** of each customer, respectively. This file is going to be used to fill the corresponding data fields (***name***, ***surname*** and ***customertype****)* in the customer struct to create a linked list of customers. For example, name of the first customer is Ali and his surname is Kaya and his customer type is **bireysel**. Similarly, second customer's name is Asli and her surname is Bulut and corresponding customer type is **kurumsal_kucuk**.

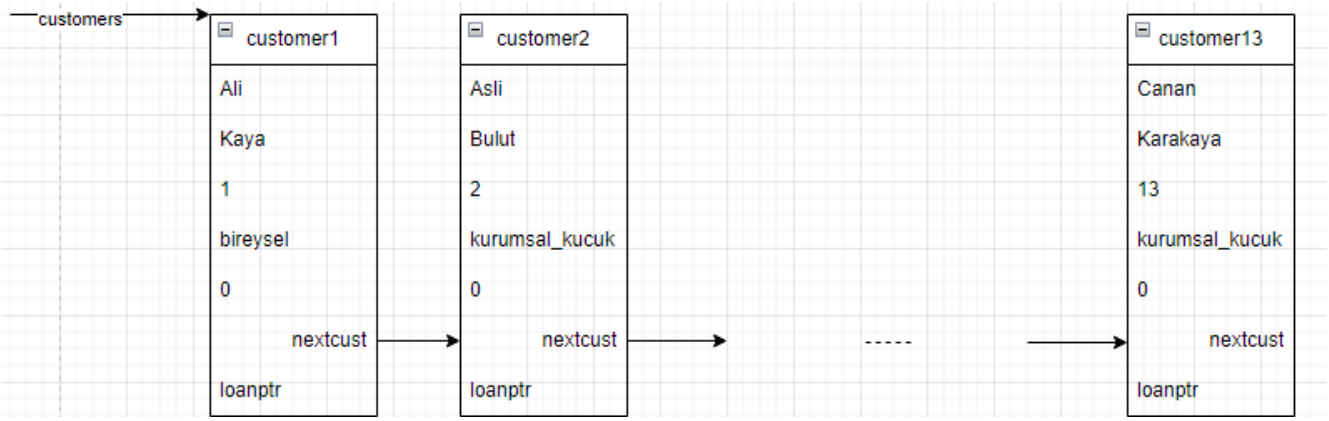
Ali Kaya bireysel

Asli Bulut kurumsal_kucuk

Mehmet Candan kurumsal_orta

…

Sample linked list:



2. option 2:

When option 2 is selected, **printCustomers** function is called and the customers linked list is traversed and *name, surname, type* and *totaldebt* values of each customers in the list are printed using printCustomers function.

Sample output:

```
please select your option :
2
###############################################################
-------------------------------------------------
1 - Ali  Kaya - type : bireysel - total debt :  0
-------------------------------------------------
2 - Asli  Bulut - type : kurumsal_kucuk - total debt :  0
-------------------------------------------------
3 - Mehmet  Candan - type : kurumsal_orta - total debt :  0
-------------------------------------------------
4 - Yasemin  Adiguzel - type : bireysel - total debt :  0
-------------------------------------------------
5 - Veli  Demir - type : kurumsal_buyuk - total debt :  0
-------------------------------------------------
6 - Suzan  Celik - type : kurumsal_sirketici - total debt :  0
-------------------------------------------------
7 - Yeliz  Kara - type : bireysel_vip - total debt :  0
-------------------------------------------------
8 - Tuna  Yildiz - type : kurumsal_kucuk - total debt :  0
-------------------------------------------------
9 - Polat  Caliskan - type : kurumsal_sirketici - total debt :  0
-------------------------------------------------
10 - Berkecan  Yildirim - type : kurumsal_orta - total debt :  0
-------------------------------------------------
11 - Demet  Avci - type : kurumsal_orta - total debt :  0
-------------------------------------------------
12 - Derya  Polat - type : bireysel - total debt :  0
-------------------------------------------------
13 - Canan  Karakaya - type : kurumsal_kucuk - total debt :  0
```

3. option 3:

When option 3 is selected, for each customer, a linked list of loans withdrawn by the customer will be created by calling **readLoans** function. To create the linked list, **loanptr** pointer of each customer will be used and members of the list will be type of ***loan struct***. Loan details of each customer is available in the **loans.txt** file and the linked list will be created using this file. The list should be **sorted by *processdate* of the loans (from oldest to newest)**. Therefore; when a new loan is read from the file it should be inserted into the correct position in the loan linked list of the customer according to its *processdate*. Please, note that the file is not sorted neither by name of the customers nor by *processdate*. Id of a loan (*loanid*) is concatenation of the id of the customer, letter L and order of the loan in the linked list. For example, 1L1 denotes first loan of the first customer and 1L2 denotes the second loan of the first customer. Similarly, 5L4 denotes the fourth loan of the fifth customer. Each loan in the linked list is connected to the next loan with ***nextloan*** pointer.

loans.txt

This file contains the information about the loans withdrawn by the customers. **Name** and **surname** of the customer**,** **type of the loan**, **total amount** of the loan**,** **number of installments** that are going to be created for the loan and **date** of the loan are given in the input file. For example, customer Canan Karakaya withdrawn loan on 10/12/2019 whose amount is $98786. Type of the loan is “kobi_destek” and 17 installments should be created for this loan. Loans withdrawn by a specific customer are kept in a linked list whose members are type of *loan struct* and this txt file is going to be used to fill the corresponding data fields (***type, totalamount, totalinstallmentnum and processdate***) of *loan structs* in the linked list.

Canan Karakaya kobi_destek 98786 17 10/12/2019

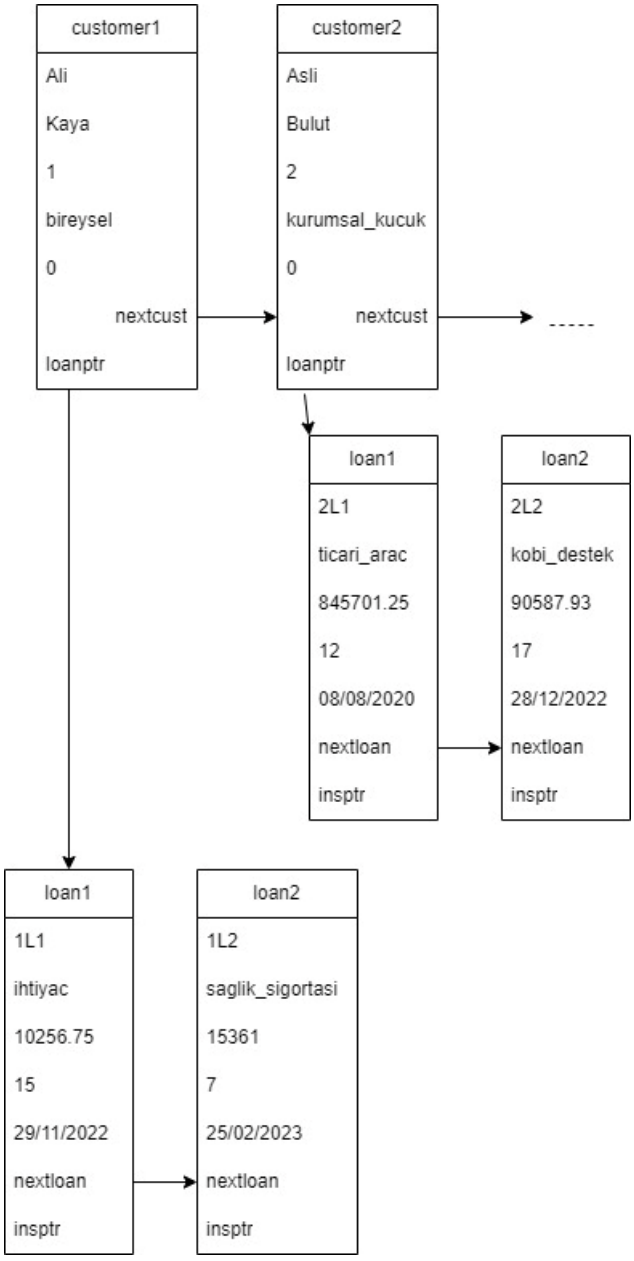
Mehmet Candan ticari_kasko 6589 3 16/04/2023

Tuna Yildiz baslangic_destek 369784 8 23/11/2022

Mehmet Candan ticari_arac 1678932 16 10/09/2022

…

Sample linked list:



4. Option 4:

If option 4 is selected, **printLoans** function is called to iterate through the linked list and print the loan related information.

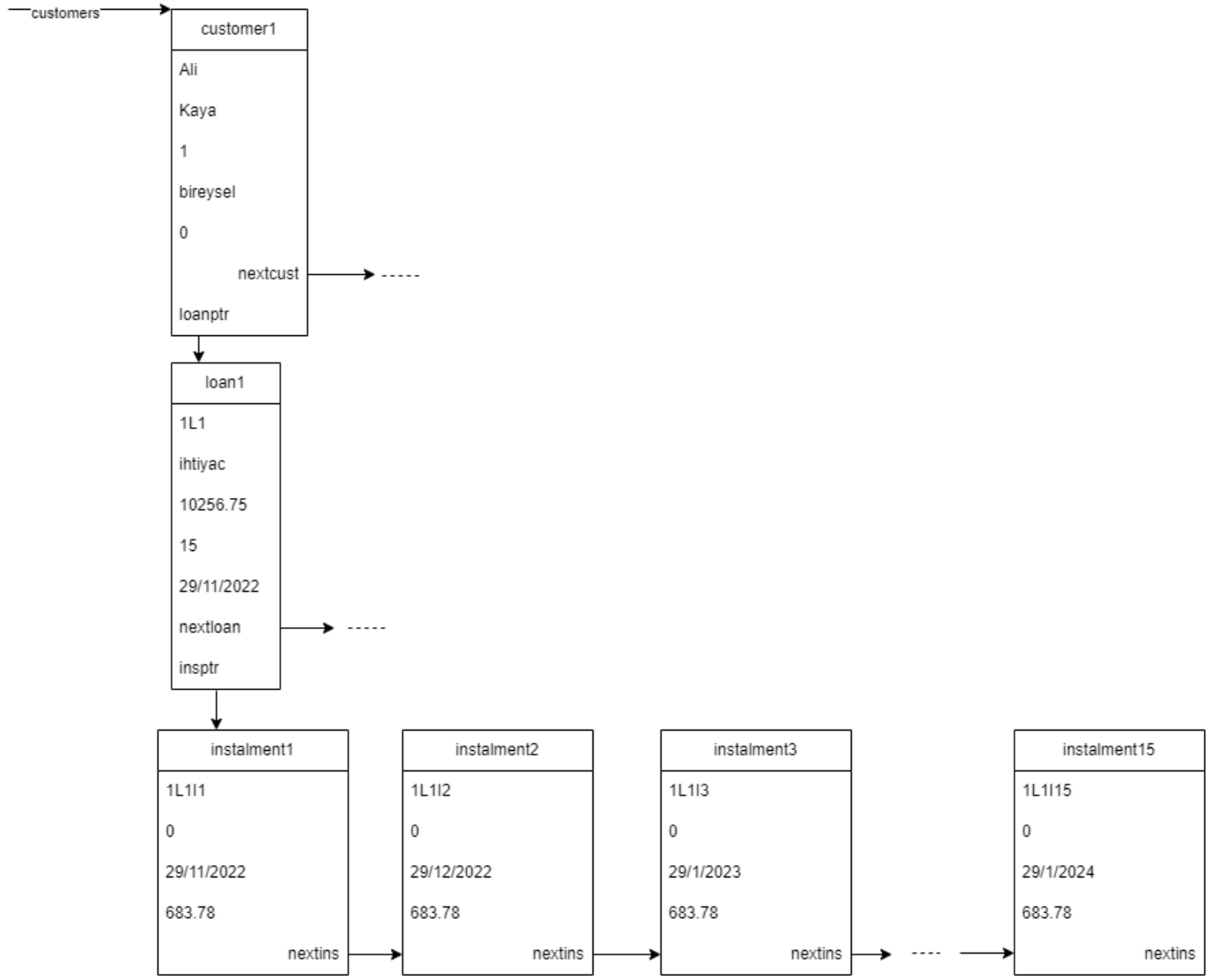
Sample partial output:

```
##################################################################
1. read customers.
2. print customers.
3. read loans.
4. print loans.
5. create installments.
6. print installments.
7. read payments.
8. find unpaid installments.
9. delete completely paid installments.
please select your option :
4
##################################################################
--------------------------------------------------
1 - Ali  Kaya - type : bireysel - total debt :  0
    1L1 : ihtiyac - 10256.75 - 29/11/2022 - 15
    1L2 : saglik_sigortasi - 15361.00 - 25/02/2023 - 7
--------------------------------------------------
2 - Asli  Bulut - type : kurumsal_kucuk - total debt :  0
    2L1 : ticari_arac - 845701.25 - 08/08/2020 - 12
    2L2 : kobi_destek - 90587.93 - 28/12/2022 - 17
--------------------------------------------------
3 - Mehmet  Candan - type : kurumsal_orta - total debt :  0
    3L1 : ticari_arac - 1678932.00 - 10/09/2022 - 16
    3L2 : ticari_kasko - 6589.00 - 16/04/2023 - 3
--------------------------------------------------
4 - Yasemin  Adiguzel - type : bireysel - total debt :  0
    4L1 : konut - 5998764.50 - 01/02/2021 - 27
    4L2 : bireysel_emeklilik - 20896.13 - 21/07/2021 - 9
--------------------------------------------------
5 - Veli  Demir - type : kurumsal_buyuk - total debt :  0
    5L1 : yatirim - 152745.98 - 10/12/2022 - 5
    5L2 : ticari_kasko - 17921.00 - 12/12/2022 - 5
    5L3 : calisan_saglik_sigortasi - 15410.00 - 15/12/2022 - 5
    5L4 : ticari_trafik_sigortasi - 4158.03 - 02/04/2023 - 4
    5L5 : isyeri_alimi - 6874581.00 - 03/04/2023 - 12
--------------------------------------------------
6 - Suzan  Celik - type : kurumsal_sirketici - total debt :  0
    6L1 : elektronik - 15426.87 - 12/12/2020 - 9
    6L2 : kasko - 4298.77 - 15/06/2021 - 3
    6L3 : trafik_sigortasi - 7863.00 - 01/01/2022 - 3
    6L4 : yatirim - 63512.31 - 15/10/2022 - 17
--------------------------------------------------
7 - Veliz  Kara - type : bireysel_vip - total debt :  0
```

5. option 5:

This option automatically creates installments of each loans of each customer calling **createInstallments** function. Installments of each loan will be kept as a linked list using ***insptr*** pointer. Each installment is connected to the next installment in the linked list with ***nextins*** pointer. Id of an installment is concatenation of the id of the loan, letter I and order of the installment in the list. For example, id 3L2I1 denotes the first installment of the second loan of the third customer. In addition, installments are inserted into lists in a sorted way (from oldest to newest). ***ispaid*** value is initially equal to 0. ***installmentdate*** of the first installment is the same as the *processdate* of the loan. *Installmentdate* of the second installment is the date 1 month later of the *processdate*. Similarly, *installmentdate* of the third installment is the date 2 months later of the *process date* and so on. You can assume that each month consists of 30 days. ***amount*** of the installment is found by dividing *totalamount* by *totalinstallmentnum (amount = totalamount / totalinstallmentnum).*

Sample linked list:



6. option 6:

This option calls the **printInstallments** function which iterates through all installments and prints the installments. If the *ispaid* value is 0, "**To be Paid"** will be printed. If an installment is paid (*ispaid=1*), **"Paid"** will be printed. If an installment is delayed and not paid (*ispaid* is a value other than 0 and 1), **"Delayed Payment"** will be printed.

Sample partial output:

```
please select your option :
6
############################################################
------------------------------------------------------
1 - Ali  Kaya - type : bireysel - total debt :  0
    1L1 : ihtiyac - 10256.75 - 29/11/2022 - 15
        1L1I1 -> 29/11/2022 - 683.78  - To be Paid
        1L1I2 -> 29/12/2022 - 683.78  - To be Paid
        1L1I3 -> 29/1/2023 - 683.78  - To be Paid
        1L1I4 -> 29/2/2023 - 683.78  - To be Paid
        1L1I5 -> 29/3/2023 - 683.78  - To be Paid
        1L1I6 -> 29/4/2023 - 683.78  - To be Paid
        1L1I7 -> 29/5/2023 - 683.78  - To be Paid
        1L1I8 -> 29/6/2023 - 683.78  - To be Paid
        1L1I9 -> 29/7/2023 - 683.78  - To be Paid
        1L1I10 -> 29/8/2023 - 683.78  - To be Paid
        1L1I11 -> 29/9/2023 - 683.78  - To be Paid
        1L1I12 -> 29/10/2023 - 683.78  - To be Paid
        1L1I13 -> 29/11/2023 - 683.78  - To be Paid
        1L1I14 -> 29/12/2023 - 683.78  - To be Paid
        1L1I15 -> 29/1/2024 - 683.78  - To be Paid
    1L2 : saglik_sigortasi - 15361.00 - 25/02/2023 - 7
        1L2I1 -> 25/02/2023 - 2194.43  - To be Paid
        1L2I2 -> 25/3/2023 - 2194.43  - To be Paid
        1L2I3 -> 25/4/2023 - 2194.43  - To be Paid
        1L2I4 -> 25/5/2023 - 2194.43  - To be Paid
        1L2I5 -> 25/6/2023 - 2194.43  - To be Paid
        1L2I6 -> 25/7/2023 - 2194.43  - To be Paid
        1L2I7 -> 25/8/2023 - 2194.43  - To be Paid
------------------------------------------------------
2 - Asli  Bulut - type : kurumsal_kucuk - total debt :  0
    2L1 : ticari_arac - 845701.25 - 08/08/2020 - 12
        2L1I1 -> 08/08/2020 - 70475.10  - To be Paid
        2L1I2 -> 8/9/2020 - 70475.10  - To be Paid
        2L1I3 -> 8/10/2020 - 70475.10  - To be Paid
        2L1I4 -> 8/11/2020 - 70475.10  - To be Paid
        2L1I5 -> 8/12/2020 - 70475.10  - To be Paid
        2L1I6 -> 8/1/2021 - 70475.10  - To be Paid
        2L1I7 -> 8/2/2021 - 70475.10  - To be Paid
        2L1I8 -> 8/3/2021 - 70475.10  - To be Paid
        2L1I9 -> 8/4/2021 - 70475.10  - To be Paid
        2L1I10 -> 8/5/2021 - 70475.10  - To be Paid
        2L1I11 -> 8/6/2021 - 70475.10  - To be Paid
        2L1I12 -> 8/7/2021 - 70475.10  - To be Paid
```

7. option 7:

When option 7 is selected, **readPayments** function is called. This function call provides reading the payments.txt file which contains which installment are paid for which loan. Moreover, it updates the *ispaid* field as 1 of the paid installments read from the file.

Payments.txt:

This file contains payments of the customers. Each payment is listed with the **id** of the loan and **which** installment of the loan is paid. For example, 9L1 ALL means all installments of loan 9L1 are paid. In addition, 1L1 4 means fourth installment of loan 1L1 is paid. Similarly, 1L1 1 means first installment of loan 1L1 is paid. Data in the file is not sorted and there can be some installments that are not included in the input file which means that they are not paid.

…

9L1 ALL

…

1L1 4

…

1L1 1

…

Sample partial output:

```
please select your option :
7
############################################################


############################################################
1. read customers.
2. print customers.
3. read loans.
4. print loans.
5. create installments.
6. print installments.
7. read payments.
8. find unpaid installments.
9. delete completely paid installments.
please select your option :
6
############################################################
------------------------------------------------
1 - Ali  Kaya - type : bireysel - total debt :  0
    1L1 : ihtiyac - 10256.75 - 29/11/2022 - 15
        1L1I1 -> 29/11/2022 - 683.78  - Paid
        1L1I2 -> 29/12/2022 - 683.78  - To be Paid
        1L1I3 -> 29/1/2023 - 683.78  - Paid
        1L1I4 -> 29/2/2023 - 683.78  - Paid
        1L1I5 -> 29/3/2023 - 683.78  - To be Paid
        1L1I6 -> 29/4/2023 - 683.78  - To be Paid
        1L1I7 -> 29/5/2023 - 683.78  - To be Paid
        1L1I8 -> 29/6/2023 - 683.78  - To be Paid
        1L1I9 -> 29/7/2023 - 683.78  - To be Paid
        1L1I10 -> 29/8/2023 - 683.78  - To be Paid
        1L1I11 -> 29/9/2023 - 683.78  - To be Paid
        1L1I12 -> 29/10/2023 - 683.78  - To be Paid
        1L1I13 -> 29/11/2023 - 683.78  - To be Paid
        1L1I14 -> 29/12/2023 - 683.78  - To be Paid
        1L1I15 -> 29/1/2024 - 683.78  - To be Paid
    1L2 : saglik_sigortasi - 15361.00 - 25/02/2023 - 7
        1L2I1 -> 25/02/2023 - 2194.43  - Paid
        1L2I2 -> 25/3/2023 - 2194.43  - Paid
        1L2I3 -> 25/4/2023 - 2194.43  - To be Paid
        1L2I4 -> 25/5/2023 - 2194.43  - To be Paid
```

8. option 8:

When option 8 is selected, **findUnpaidInstallments** function is called. This function takes a date from the user and iterates through installments and finds the installments that should have been paid before the date entered by the user. The installments that are not paid before this date are evaluated as "**Delayed Payment**" and *ispaid* value is updated with a value other than 0 and 1. Then, for each customers with delayed payments total amount of unpaid installments (***debt***) and number of delayed installments are printed.

Sample output:

```
1. read customers.
2. print customers.
3. read loans.
4. print loans.
5. create installments.
6. print installments.
7. read payments.
8. find unpaid installments.
9. delete completely paid installments.
please select your option :
8
######################################################################
please enter a date:
01/06/2023
Ali Kaya : Debt 7123.99 Number of Delayed Installments 6
Asli Bulut : Debt 21314.81 Number of Delayed Installments 4
Mehmet Candan : Debt 419733.00 Number of Delayed Installments 4
Veli Demir : Debt 706870.65 Number of Delayed Installments 9
Suzan Celik : Debt 3736.02 Number of Delayed Installments 1
Tuna Yildiz : Debt 92446.00 Number of Delayed Installments 2
Berkecan Yildirim : Debt 37706.46 Number of Delayed Installments 1
Demet Avci : Debt 8988.99 Number of Delayed Installments 3
Derya Polat : Debt 871.78 Number of Delayed Installments 1
Canan Karakaya : Debt 2293.30 Number of Delayed Installments 1


######################################################################
1. read customers.
2. print customers.
3. read loans.
```

9. option 9:

When option 9 is selected, **deletePaidInstallments** function is called and all loans whose all installments are paid deleted from the related linked list.

Sample partial output:

```
--------------------------------------------------
3 - Mehmet  Candan - type : kurumsal_orta - total debt : 419733
    3L1 : ticari_arac - 1678932.00 - 10/09/2022 - 16
        3L1I1 -> 10/09/2022 - 104933.25  - Paid
        3L1I2 -> 10/10/2022 - 104933.25  - Delayed Payment
        3L1I3 -> 10/11/2022 - 104933.25  - Delayed Payment
        3L1I4 -> 10/12/2022 - 104933.25  - Paid
        3L1I5 -> 10/1/2023 - 104933.25  - Paid
        3L1I6 -> 10/2/2023 - 104933.25  - Paid
        3L1I7 -> 10/3/2023 - 104933.25  - Paid
        3L1I8 -> 10/4/2023 - 104933.25  - Delayed Payment
        3L1I9 -> 10/5/2023 - 104933.25  - Delayed Payment
        3L1I10 -> 10/6/2023 - 104933.25  - To be Paid
        3L1I11 -> 10/7/2023 - 104933.25  - To be Paid
        3L1I12 -> 10/8/2023 - 104933.25  - To be Paid
        3L1I13 -> 10/9/2023 - 104933.25  - To be Paid
        3L1I14 -> 10/10/2023 - 104933.25  - To be Paid
        3L1I15 -> 10/11/2023 - 104933.25  - To be Paid
        3L1I16 -> 10/12/2023 - 104933.25  - To be Paid
--------------------------------------------------
4 - Yasemin  Adiguzel - type : bireysel - total debt :  0
        no loan
--------------------------------------------------
5 - Veli  Demir - type : kurumsal_buyuk - total debt : 706871
    5L1 : yatirim - 152745.98 - 10/12/2022 - 5
        5L1I1 -> 10/12/2022 - 30549.20  - Paid
        5L1I2 -> 10/1/2023 - 30549.20  - Delayed Payment
        5L1I3 -> 10/2/2023 - 30549.20  - Delayed Payment
        5L1I4 -> 10/3/2023 - 30549.20  - Delayed Payment
        5L1I5 -> 10/4/2023 - 30549.20  - Delayed Payment
    5L2 : ticari_kasko - 17921.00 - 12/12/2022 - 5
        5L2I1 -> 12/12/2022 - 3584.20  - Paid
        5L2I2 -> 12/1/2023 - 3584.20  - Paid
        5L2I3 -> 12/2/2023 - 3584.20  - Delayed Payment
        5L2I4 -> 12/3/2023 - 3584.20  - Delayed Payment
        5L2I5 -> 12/4/2023 - 3584.20  - Delayed Payment
    5L4 : ticari_trafik_sigortasi - 4158.03 - 02/04/2023 - 4
        5L4I1 -> 02/04/2023 - 1039.51  - Paid
        5L4I2 -> 2/5/2023 - 1039.51  - Delayed Payment
        5L4I3 -> 2/6/2023 - 1039.51  - To be Paid
        5L4I4 -> 2/7/2023 - 1039.51  - To be Paid
```

Implementation Details:

1. You are supposed to use linked lists of given structs. Therefore, arrays cannot be used instead of linked lists.

2. You are not allowed to change/modify/add/delete any data fields of the given structs. In addition, it is not allowed to add other kind of structs.

3. Usage of global pointers and global variables is not allowed. Moreover, static and constant variables are not allowed.

4. Menu will be shown to the user again and again until 0 is entered as the option.

5. All file operations, linked list creation/deletion/modification/insertion etc. operations will be performed by function calls. Therefore, the main function will be responsible for just calling required function calls and displaying the menu.

6. It is not allowed to change/modify format of the input files.

7. Data read from input files will be stored in the corresponding data fields of the structs of the linked lists.

8. Functions in options 2,4 and 6 will print the values iterating over the linked lists and printing the values inside the data fields of the structs.

9**. Your output format should be exactly the same format as the one given in the sample outputs**.

10. Please print the output both on the console & output.txt file.

11. Please use relative addressing for input & output files.

Submission Details:

1. This assignment will be done in groups of 2 people. Each group must elect a group representative.

2. Only the group representative will submit the homework on behalf of the group.

3. Please submit your **source code, report and output.txt** file as a single .zip file. Name of the file will be the ids of the group members (i.e, project1_150111111_150222222.zip)

4. Write a report and submit as a .pdf file

i. indicate which options are completed

ii. indicate which options are not completed

iii. indicate which parts work as expected/gives correct output

iv. indicate which options do not work/unexpected output

v. Provide screenshot of output for each option.,

vi. Write explanations about the functions you have written. Include the purpose of the function and how it does which task.

vii. Explain where and how the linked lists are created and elements of a linked list are deleted. If a linked list has a sorted structure, please explain where and how it is sorted.