

Report on the bank account management program

-This report contains description of our code, all algorithms used, and a high level illustration using Pseudocode.

-We also added a user manual to help you use our program easily.

-Sample runs were also added after each section to help you cope with the program.

Prepared by

- **Youssef Amr Keshk 8852**
- **Zeyad Essam Haridy 8936**
- **Andrew Ayman Makrem 8575**
- **Ahmed Emad Ahmed 8600**

At the beginning of the code some global variables were declared, so they could be used in all functions.

First, two structures were defined to represent all account's fields

1- "account" containing the following variables: string "account number", string "name", string "mobile", string "email", date_opened "date".

2- "date_opened" containing the following variables: array of characters "months" and the array of characters "year"

Second, six global variables were declared:

1- pointer to account "**Accounts**"

2- integer variable "**NUM_ofaccounts**" representing the total number of accounts saved in "Accounts"

3- string variable **username**

4- string variable **password**

5-integer **indexofaccount** //which will be used in **edit_fileforaccount()** function

6- integer **indexofreceiver** //which will be used in **edit_fileforaccount()** function

Third, five functions were made, so they would be used in most of the main functions

1-NUM-validation

This function takes a pointer to a character as an argument and returns an integer. The purpose of this function is to check whether the account number entered by the user is unique or has a match. It uses a **Linear Search** algorithm to compare each account number in "Accounts" with the entered account number using **strcmp()**.

In case strcmp() returned 0

The function will return "i" which is the index of the entered account in "Accounts"

In case strcmp() did not return 0

The process will be repeated after incrementing the index "i" by 1 in the for() loop.

In case "i" reached the maximum value

If **strcmp()** never returned 0, the function will return -1 as indicating that no matches were found

Pseudocode representing the linear search algorithm used in NUM_validation

```
function num_validation(x):
    flag = 0
    for i from 0 to NUM_ofaccounts - 1:
        flag = strcmp(x, Accounts[i].acc_num)
        if flag == 0:
            return i
    return -1
```

2-Name-validation

This function takes a pointer to a character as an argument and returns an integer. The purpose of this function is to validate the new account name entered by the user in the functions **add_Account()** and **modify()** and also to convert name into the same format as all other names. Furthermore, **toupper(newname[0])** is used to convert the first letter into uppercase. Then, **strlen()** is used to calculate the length of the of the account name and a **for()** loop is used to run through each character is the new name:

1-if the character is a space character, a variable “space”(initially set to 0) will be incremented by 1 and the next character will be converted to uppercase using **toupper(newname[i + 1])** ;to store all names in the same format.

2- if the condition is false ,**isalpha()** will be used to check that the character is a letter, where the program will print an error message “Error account name should consist of only letters” if it return 1 and then the function will return 0, indicating that the name is invalid.

*When the loop ends, “spaces” will be checked if it’s equal to 1, the function will return 1, else the program will print “Account name should consist of 2 names” and return 0.

3-Email-validation

This function takes a pointer to a character as an argument and returns an integer. The purpose of this function is to validate the new email address entered by the user in the functions **add_Account()** and **modify()**. The purpose of this function is make sure the email has exactly 1 ‘@’ and at least 1 dot. A for loop is used to run through each character, if the character is equal to ‘@’ the variable “atcount”(initially set to 0) will be incremented by 1, else if the character is equal to ‘.’ the variable “dotcount”(initially set to 0) will be incremented by 1. At the end of the

loop if one of the desired conditions is false, function will return 0 and if all conditions are true it will return 1.

4-month-Name

This function takes a pointer to an account (which is the month as a number) as an argument and returns a pointer to a character (which is the month as a name).

5-edit-fileforaccount

The purpose of this function is to create or edit the files holding the accounts transactions' history. This function takes two arguments, the first is a double representing the amount in \$ and the second represents the type of the transaction.

At the beginning, **sprintf()** function is used to copy the account number next to the file type(.txt) into the declared variable "string". Then:

In case transaction type is equal to 1

fopen() is used to open the account's file in append mode and **fprintf()** is used to print the transaction done in the file in this format "Withdraw by (ammount)\$", then **fclose()** is used to close to opened file.

In case transaction type is equal to 2

fopen() is used to open the account's file in append mode and **fprintf()** is used to print the transaction done in the file in this format "Deposite by (ammount)\$", then **fclose()** is used to close to opened file

In case transaction type is equal to 3

fopen() is used to open the sender account's file in append mode and **fprintf()** is used to print the transaction done in the file in this format "Transfer from account by (ammount)\$", then **fclose()** is used to close to opened file.

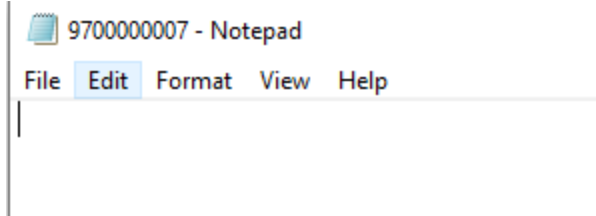
Then the same steps are repeated with the receiver account's file.

In case transaction type is equal to 0

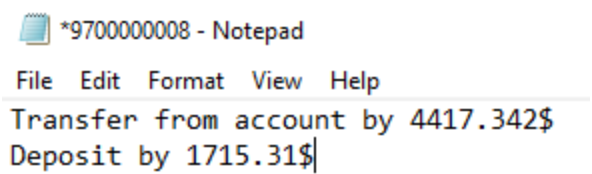
fopen() is then used in write mode to create a file for the newly added account.**fclose()** is used to close to opened file

In case transaction type is equal to -1

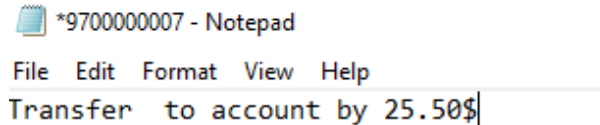
remove() will be used to delete the file of the deleted account



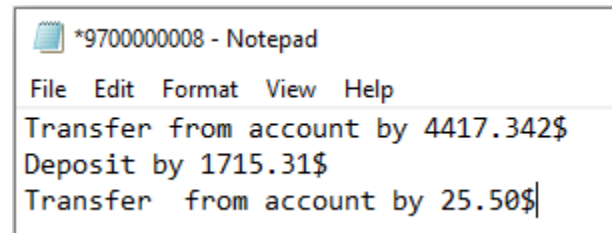
9700000007.txt befor calling edit_fileforaccount()



9700000008.txt befor calling edit_fileforaccount()



9700000007.txt after calling edit_fileforaccount()



9700000008.txt after calling edit_fileforaccount()

Now for the main functions:

1-Login

At the beginning of the program in the **main** function, an **initial_Menu()** function is called, where the user will be asked to enter either 'Q' to quit or 'L' to login. **toupper()** function was used to handle lower and upper cases.

In case user entered 'L'



LOGIN() function will be called, where then the user will be prompted to enter username and password and both will be assigned to the variables Username and Password.

To validate the user's inputs a **Linear search** algorithm. First, **fopen()** function is used to open users.txt file in read mode. Second, **fscanf()** function was used to read the first line in **users.txt**, where it will stop scanning after first whitespace, and then assign the username to hold_username. The same function was used for the second time, where it will now stop after reaching '\n', and then assign the password to hold_password.

strcmp() function is now used twice to compare hold_username with Username and hold_password with Password.

If both were found identical and **strcmp** returned 0 for the two comparisons, **fclose()** will be used to close the opened file and **LOGIN()** will return 1. If one of the conditions was not true, process will iterate in a while() loop, until both conditions are achieved. If the conditions were



never archived, the while loop will stop when end of file is reached and **feof()** returns 0, then a message will be printed saying “INVALID username and password” and process returns 0.

| | |
|---|--|
|  Select C:\Users\Youssef\Deskt |  Select C:\Users\Youssef\Desktop\zeyad\bin\ |
| Enter L to LOGIN Enter Q to QUIT 1 Username: ziad.ali Password: 123abc_ | Enter L to LOGIN Enter Q to QUIT 1 Username: ali.ziad Password: 1a2b3c INvalid username and password Enter L to LOGIN Enter Q to QUIT |

In **initial_Menu()**, the return of **login()** will be assigned to a variable named flag, if flag was equal to 1, **initial_Menu** will return 1, else the whole process will be repeated from the start.

In case user entered 'Q'

initial_Menu will return 0 to the main.

| | |
|--|---|
|  Select C:\Users\Youssef\Deskt |  Select C:\Users\Youssef\Desktop\zeyad\bin\Debug\zeyad.exe |
| Enter L to LOGIN Enter Q to QUIT q | Program closed. Have a great day! Process returned 0 (0x0) execution time : 14.954 s Press any key to continue. |

2-Load

Firstly, **fopen()** is used to open the file accounts.txt in read mode, the total number of accounts will be calculated by incrementing the global variable NUM_ofaccounts(initially set to zero) by 1 each time a new line character(\n) is scanned by **fgetc()** this process will be repeated in a while loop until EOF is reached. When the loop stops, NUM_ofaccounts is incremented by 1 to add the last account in the file. Then **rewind()** is used to reset the pointer's position to the start of the file.

Sencondly, the global pointer to account Accounts” is dynamically allocated using **malloc()** which allocates contiguous bytes of memory (on the heap).

Thirdly, a `for()` loop was used to store all the data from `accounts.txt` in `Accounts`. A simple algorithm was used to copy the data line by line from the file.

1- **`fgets()`** reads the first line line in the file and assign it to the variable `hold_string`.

2-**`strtok(hold_string, “,”)`** is used to break `hold_string` into a sequence fo tokens where ‘,’ is the delimiter for separating tokens.

3- **`strcpy()`** is then used to copy to the content of pointer returned by **`strtok()`** (inculding the null character) into `Accounts[i].acc_num`.

4-step 2 and 3 are repeated with all `Accounts` fields with the exception that **`strtok()`** in each subsequent call in be called by **`strtok(NULL , “,”)`** to continue tokenizing the same string.

5- when all fields are done, the loop iterates and repeates all the previous steps until the index reaches `NUM_ofaccounts`.

At the end, **`fclose()`** was used to close the `accounts.txt` file.

3-(Query) Search

At the beginning, the user is prompted for the account number and 2 steps of validation are made on this number:

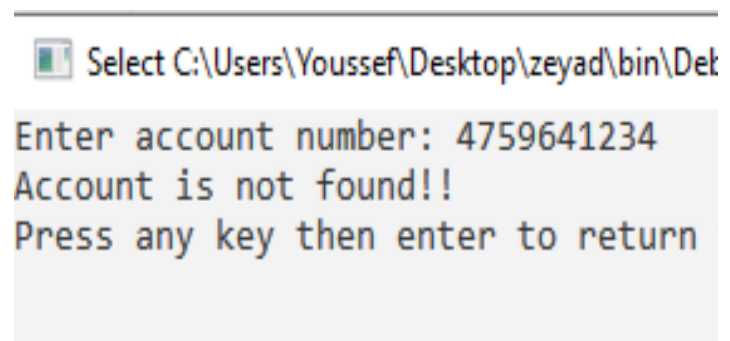
1-**`strlen()`** was used to check that account number consists of exactly 10 digits. If the condition is false, the program prints “INVALID account number” and repeats the steps from the start.

2-**`NUM_validation()`** was called to check wheather the number has a match or not. The program prints “Account is not found” if it returned -1.

Lastly, the return of **`NUM_validation()`** is assigned to the integer variable `<index>` and then the program prints all the fields’ data saved at this index in order. (**`month_Name()`** is called to convert the representation the month from a number a name)



```
Select C:\Users\Youssef\Desktop\zeyad\bin\Debug\zeyad.exe
Enter account number: 5798431478
Account Number: 5798431478
Name: Mazen yousef
E-mail: maz.youssef@yahoo.com
Balance: 46784.24
Mobile: 01114797423
Date Opened: January 2023
```



```
Select C:\Users\Youssef\Desktop\zeyad\bin\De
Enter account number: 4759641234
Account is not found!!
Press any key then enter to return
```

4-Advanced Search

First, the user is asked to enter a keyword, then to handle lower and upper cases, the two char variables <keyword_lowercase> and <name_lowercase> were declared.

Second, for() loop was used to convert each character in the keyword to a lowercase character by calling **tolower(keyword[j])** and use **strcpy()** to copy it in <keyword_lowercase>.

Third, for() loop was used to check every account in <Accounts> through a series of steps:

1-for() loop was used to convert each character in Accounts[i].name to a lowercase character by calling **tolower(Accounts[i].name[j])** and use **strcpy()** to copy it in <name_lowercase>.

2-**strstr(name_lowercase, keyword_lowercase)** used to check whether keyword_lowercase is part of name_lowercase

3-if **strstr()** did not return NULL then <keyword_lowercase> is part of <name_lowercase> and function will print all the fields' data saved at this index in order and set the variable flag to 1 (initially set to 0).

4-Steps are repeated for all accounts, by a **Linear Search** like algorithm.

If flag is equals to 0 and no matches were found function will print "No matches are found".

```
Select C:\Users\Youssef\Desktop\zeyad\bin
Enter Keyword: youssef

Account Number: 5798431478
Name: Mazen youssef
E-mail: maz.youssef@yahoo.com
Balance: 46784.24
Mobile: 01114797423
Date Opened: January 2023

Account Number: 3265840932
Name: Youssef Keshk
E-mail: doraroxo@gmail.com
Balance: 57000.66
Mobile: 01121886991
Date Opened: February 2004
```

```
Select C:\Users\Youssef\Desktop\zeyad\bin\Debug\zeyad.exe
Enter Keyword: john
No matches are found
Press any key then enter to return to the menu.
```

5-ADD

First the user is prompted for the new account number and 4 steps of validation were made on the account number :

1-**strlen()** function was used to make sure number consists of exactly 10 digits.

2-the first character(`newaccount.acc_num[0]`) was check it's not equal to '0'.

3-for() loop is used to check that all the the new account number characters are digits, by assuring that `newaccount.acc_num[i]` lies between '0' and '9' inclusive.

*If one of the three conditions were false, a message will be printed saying "Error: Account number should consist of 10 digits" then "flag" will be set to 1.

4- **NUM_validation()** is used to check that the new account number is unique and has no matches. If it returned anything rather than -1, then account number is not unique, so "flag" will be set to 1 and a message will be printed saying "Error: Account number already exists."

*After the 4 conditions are checked, if the "flag" is equal to 1 (which means account number is invalid), "flag" will be reset to 0 and the process will repeat from the start using a do while() loop.

Furthermore, the user will be prompt field by field for the data of the new account except the account's balance where it will initially be set to 0.00\$.

1- **gets()** is used to get the name from the user including spaces and **name_validation()** was used to validate the name.

2- **strlen()** is used to make sure mobile number consists of 11 digits.

3- **email_validation()** is used to validate email address.

4- **time(NULL)** is used to returns the current time and **localtime()** converts it to a structure containing the broken-down time information.

Moreover, the user is asked to enter either 'S' to save or 'D' to discard changes. **Toupper()** is used to accept lower and upper cases. If user entered anything rather than 's' or 'd' the question will be repeated using a do while() loop.

In case user entered 'S'

1- "NUM_ofaccounts" will be incremented by 1

2- **realloc()** function will be used to reallocate the memory location of the account pointer "Accounts" to add the newly added account to Accounts.

3- the "newaccount" data will be assigned to "Accounts[NUM_of accounts -1]".

4- **sprintf()** is used to copy the time in the string Accounts[i].date

5- **save_Data()** will be called to save the changes in the accounts.txt file and program will prints "Account successfully added!"

6- the global variable "indexofaccount" will be updated with the new account's index and **edit_fileforaccount(0,0)** will be called.

In case user entered 'D'

Function will print "Changes are canceled".

```
C:\Users\Youssef\Desktop\zeyad\bin\Debug\zeyad.exe
```

```
Enter new account number: 6545469874
Error: Account number already exists.
Enter new account number: 1258746931
Enter account holder's name: Ibrahim Ali
Enter mobile number: 01151886442
Enter e-mail address: ibra.05@gmail.com
Enter (S) to save the changes and add the account or (D) to discard the changes: s
Account successfully added!
```

```
C:\Users\Youssef\Desktop\zeyad\bin\Debug\zeyad.exe
```

```
Enter new account number: 789458764524
Error: Account number should consist of 10 digits.
Enter new account number: a123bc566
Error: Account number should consist of 10 digits.
Enter new account number: 8759642387
Enter account holder's name: Essam Haridy
Enter mobile number: 010017465467
Invalid mobile number.
Enter mobile number: 010017465467
Invalid mobile number.
Enter mobile number: 01001746546
Enter e-mail address: essam@yahoo.com
Enter (S) to save the changes and add the account or (D) to discard the changes: d
Changes are canceled
```

6-Delete

First the user is prompted for the number of the account to be deleted, then some validation steps will be made:

1- **NUM_validation()** is used to check that the entered number is found and assign the index to "index". If it returned -1 the program will print "Error: Account not found" and the process will be repeated from the start using a do while() loop.

2- **atof()** is used to converted the account balance from string into float to check whether it's equal to 0 or not. If the condition is false the program will print "Deletion failed. Account has a balance greater than 0.00\$" and stop.


Moreover, the user is asked to enter either 'S' to save or 'D' to discard changes. **Toupper()** is used to accept lower and upper cases. If user entered anything rather than 's' or 'd' the question will be repeated using a do while() loop.

In case user entered 'S'


- 1- the global variable "indexofaccount" will be updated with the new account's index and **edit_fileforaccount(0,-1)** will be called.
- 2- **memset (&Accounts[index], '\0' , sizeof(account))** will be used to fill the memory with value '\0' to delete it.
- 3- "NUM_of accounts" will be decremented by 1
- 4- the pointer to account "Accounts" will be adjusted by shifting the indices after the deleted index, at which **realloc()** function will then be used to reallocate the memory location of the account pointer "Accounts" to remove the extra reserved memory block.
- 5- **save_Data()** will be called and program will print "Account deleted successfully".

In case user entered 'D'


Program will print "Deletion is canceled"

 C:\Users\Youssef\Desktop\draft\bin\Debug\draft.exe

```
Enter the bank account number to be deleted: 1475463784
Enter (S) to save changes and delete account or (D) to discard the changes: s
Account deleted successfully.
```

 C:\Users\Youssef\Desktop\draft\bin\Debug\draft.exe


```
Enter the bank account number to be deleted: 9780136019
Deletion failed. Account has a balance greater than zero.
```

 *accounts - Notepad

File Edit Format View Help

```
9700000000,Michael Jones,m.jones@gmail.com,1000,01009700000,12-2007
9700000001,Ali Ahmed,j.roberto@outlook.com,100,01115427894,12-2008
9700000002,Timothy Korman,t.korman@gmail.com,200,01009700002,12-2015
9700000003,Michael Robert,michael@yahoo.com,0.00,01009700003,11-2008
```

accounts.txt befor Delete

 *accounts - Notepad

File Edit Format View Help

```
9700000000,Michael Jones,m.jones@gmail.com,1000,01009700000,12-2007
9700000001,Ali Ahmed,j.roberto@outlook.com,100,01115427894,12-2008
9700000002,Timothy Korman,t.korman@gmail.com,200,01009700002,12-2015
```

accounts.txt after Delete

7- Modify

First the user enters the account number which is been validated, and then we get the index of the account from the function **num_validation()**, then the user is asked whether to modify the 1.name, 2.mobile number, or 3.the email address, the user should enter the number of the data to be modified and this number is validated to be between 1 and 3.

case of name:

The user is required to enter the new name. The new name is taken to the variable "newname" by **gets()**. The name is validated using the function **name_validation()**, then the user is asked if he wants to save the changes, a character is scanned which must be 'y' or 'd', and **toupper()** function is used to handle upper and lower case of the character. if the user enters 'y', the new name is modified in the struct itself by **strcpy()** which replaces the name of the account with the new name. then function **save_data()** is called to save the modification into the file. if the user enters 'd' nothing is changed.

```
C:\Users\power\Desktop\programing_1\FinalProject\Final_Project\bin\Debug\Final_Project.exe
Enter account number: 9780136019
Enter (1) to modify name
Enter (2) to modify mobile number
Enter (3) to modify email address
1
Enter the new name: ahmed mo7amed
Error account name should consist of only letters
Enter the new name: ahmed
account name should consist of 2 names
Enter the new name: ahmed mohammed
Enter (S) to save changes and confirm transactions or (D) to discard the changes: s
Modification was done successfully!
Do you want to modify any thing else?
Enter (Y) for yes or (N) for no: y
Enter (1) to modify name
Enter (2) to modify mobile number
Enter (3) to modify email address
1
Enter the new name: mohammed ahmed
Enter (S) to save changes and confirm transactions or (D) to discard the changes: d
Modification is canceled
Do you want to modify any thing else?
Enter (Y) for yes or (N) for no: n
```

case of mobile number:

Same process is repeated as in name modification but the difference is that the mobile number is validated to be 11 numbers with no characters.

```
C:\Users\power\Desktop\programing_1\FinalProject\Final_Project\bin\Debug\Final_Project.exe
Enter account number: 9700000006
Enter (1) to modify name
Enter (2) to modify mobile number
Enter (3) to modify email address
2
Enter the new mobile number: 0100000000
Error mobile number should consist of 11 numbers
Enter the new mobile number: 010000000000
Enter (S) to save changes and confirm transactions or (D) to discard the changes: s
Modification was done successfully!
Do you want to modify any thing else?
Enter (Y) for yes or (N) for no: n
```

case of email address:

Same process is repeated as in name modification, **emailvalidation()** function is used to validate the new e-mail.

```
C:\Users\power\Desktop\programing_1\FinalProject\Final_Project\bin\Debug\Final_Project.exe
Enter account number: 9700000006
Enter (1) to modify name
Enter (2) to modify mobile number
Enter (3) to modify email address
3
Enter the new email address: aaaa@b
INvalid e-mail address!
Enter the new email address: ahmed@gmail.com
Enter (S) to save changes and confirm transactions or (D) to discard the changes: d
Modification is canceled
Do you want to modify any thing else?
Enter (Y) for yes or (N) for no: n
```

At the end the user is asked if he wants to modify anything else. a character is scanned from 'y' or 'n', if the user enters 'y' the whole function is repeated again. Otherwise if 'n' is entered the user returns to the menu.

8- Withdraw

First the user is prompted for the number of the account for the transaction.

Then **NUM_validation()** is used to check that the entered number is found and assign the index to "index". If it returned -1 the program will print "Error: Account not found" and the process will be repeated from the start using a do while() loop.

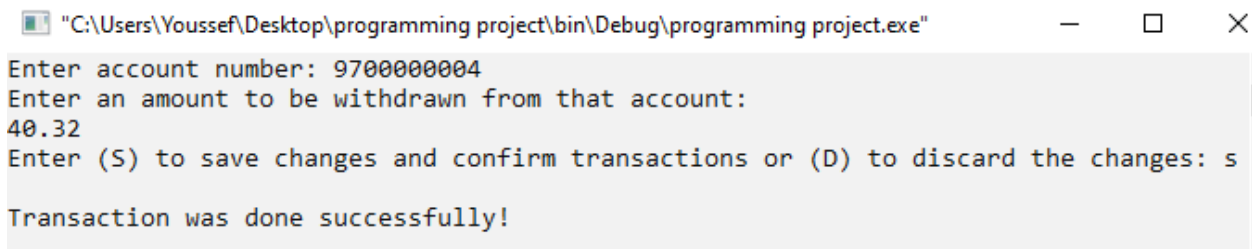
Moreover, the user is asked to enter the withdrawal amount then if this amount is greater than 10,000\$ the program will print "Error: maximum withdrawal limit is 10,000\$ per transaction" and user will be asked again to enter an amount through a do while() loop.

Furthermore, `atof(Accounts [index].balance)` is used to convert the account balance into a float to compare it with the entered amount. If the amount is greater than the account balance the program will print "Insufficient funds. Your account balance is below the withdrawal amount" and return.

Then, the user is asked to enter either 'S' to save or 'D' to discard changes. **Toupper()** is used to accept lower and upper cases. If user entered anything rather than 's' or 'd' the question will be repeated using a `do while()` loop.

In case user entered 'S'

- 1- a double variable "newbalance" will hold the result of the subtraction of the entered amount from **`atof(Accounts [index].balance)`**
- 2- **`gctv()`** function will be used to convert the double "newbalance" into a string and assign it to "Accounts [index].balance"
- 3- **`save_Data()`** will be called to save the changes in the accounts.txt file and program will prints "Transaction was done successfully!"
- 4- the global variable "indexofaccount" will be updated with the withdrawal account's index and **`edit_fileforaccount(amount, 1)`** will be called.



```
"C:\Users\Youssef\Desktop\programming project\bin\Debug\programming project.exe"
Enter account number: 9700000004
Enter an amount to be withdrawn from that account:
40.32
Enter (S) to save changes and confirm transactions or (D) to discard the changes: s
Transaction was done successfully!
```

9- Deposit

First the user is prompted for the number of the account for the transaction.

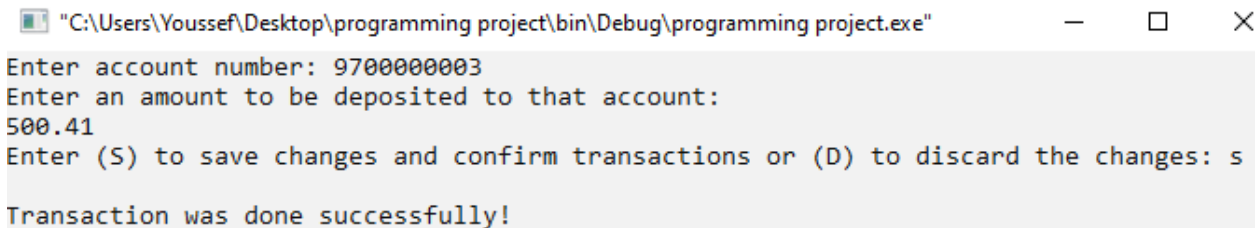
Then **`NUM_validation()`** is used to check that the entered number is found and assign the index to "index". If it returned -1 the program will print "Error: Account not found" and the process will be repeated from the start using a `do while()` loop.

Moreover, the user is asked to enter the deposit amount then if this amount is greater than 10,000\$ the program will print "Error: maximum deposit amount is 10,000\$ per transaction" and user will be asked again to enter an amount through a `do while()` loop.

Then, the user is asked to enter either 'S' to save or 'D' to discard changes. **Toupper()** is used to accept lower and upper cases. If user entered anything rather than 's' or 'd' the question will be repeated using a do while() loop.

In case user entered 'S'

- 1- a double variable "newbalance" will hold the result of the addition of the entered amount with **atof (Accounts [index].balance)**
- 2- **gctv()** function will be used to convert the double "newbalance" into a string and assign it to "Accounts [index].balance"
- 3- **save_Data()** will be called to save the changes in the accounts.txt file and program will prints "Transaction was done successfully!"
- 4- the global variable "indexofaccount" will be updated with the deposition account's index and **edit_fileforaccount(amount, 2)** will be called.



```
"C:\Users\Youssef\Desktop\programming project\bin\Debug\programming project.exe"
Enter account number: 9700000003
Enter an amount to be deposited to that account:
500.41
Enter (S) to save changes and confirm transactions or (D) to discard the changes: s
Transaction was done successfully!
```

10- Transfer

First the user is prompted for the number of account of the sender and the number of account of the receiver for the transaction.

Then **NUM_validation()** is used to check that the entered account numbers are found and assign the index to "index". If it returned -1 the program will print "Error: Account not found" and the process will be repeated from the start using a do while() loop.

Moreover, the user is asked to enter the transfer amount, then if this amount is greater than 10,000\$ the program will print "Error: maximum withdrawal limit is 10,000\$ per transaction" and user will be asked again to enter an amount through a do while() loop.

Furthermore, **atof (Accounts [index].balance)** is used to convert the account balance into a float to compare it with the entered amount. If the amount is greater than the account balance the program will print "Insufficient funds. Your account balance is below the withdrawal amount" and return.

Then, the user is asked to enter either 'S' to save or 'D' to discard changes. **Toupper()** is used to accept lower and upper cases. If user entered anything rather than 's' or 'd' the question will be repeated using a do while() loop.

In case user entered 'S'

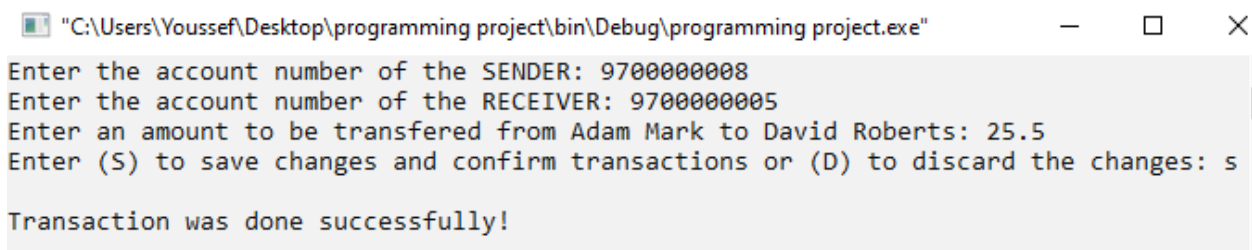
1- a double variable "newbalanceofsender" will hold the result of the subtraction of the entered amount from **atoi (Accounts [index_of_sender].balance)**

2- a double variable "newbalanceofreceiver" will hold the result of the addition of the entered amount with **atoi (Accounts [index_of_receiver].balance)**

2- **gctv()** function will be used to convert the double "newbalanceofsender" into a string and assign it to "Accounts [index_of_sender].balance" and also to convert the double "newbalanceofreceiver" into a string and assign it to "Accounts [index_of_receiver].balance"

3- **save_Data()** will be called to save the changes in the accounts.txt file and program will prints "Transaction was done successfully!"

4- the global variable "indexofaccount" will be updated with the withdrawal account's index, also the global variable "indexofreceiver" will be updated with the deposition account's index and **edit_fileforaccount(amount, 3)** will be called.



```
"C:\Users\Youssef\Desktop\programming project\bin\Debug\programming project.exe"
Enter the account number of the SENDER: 9700000008
Enter the account number of the RECEIVER: 9700000005
Enter an amount to be transfered from Adam Mark to David Roberts: 25.5
Enter (S) to save changes and confirm transactions or (D) to discard the changes: s
Transaction was done successfully!
```

11-Report

First the user is prompted for the number of the account to be reported

Then **NUM_validation()** is used to check that the entered account number is found and assign the index to "index". If it returned -1 the program will print "Error: Account not found" and the process will be repeated from the start using a do while() loop.

1- **sprintf()** function is used to copy the account number next to the file type(.txt) into the declared variable "string".

2- **fopen()** is used to open the account's file in read mode.

If fopen() returned NULL

Function will print "NO transactions were made on this account"

If fopen() did not return NULL

The total number of transactions on this account will be calculated using **fgetc()** will get a character from the file, if this character is a new line character then the variable "n"(initially set to 0) will be incremented by 1

This process will be repeated until **feof()** returns 1, the **rewind()** will be used to reset the pointer position to the start of the file.

In case "n" is greater than 5

1- **fgets()** will be used to copy line by line from the file and assign it to the string available "line"

2- then function will print the content of "line"

This process will be repeated using a for() loop from index 0 till n -1

In case "n" is smaller than or equal to 5

Same process will be done, by it will end when EOF is reached.

```
"C:\Users\Youssef\Desktop\programming project\bi...
Enter account number: 9700000001
The last 5 transactions made on this account are:
Transfer from account by 400.50$

Deposit by 7741.21$

Transfer from account by 150.50$

Transfer from account by 250.00$
```

12- print:

A new pointer sort is dynamically allocated using **malloc()** to allocate contiguous bytes of memory (on the heap). It is used to receive the data sorted from the function and avoid the data being deleted before receiving it. The user is asked if he wants the data sorted by name, balance, or the date. The user enters a number from 1 to 3 indicating the sort type and this number is validated. **getchar()** is used to take the '\n'. then a function is called to sort the data depending on the number entered whether it is **sortbyname()**, **sortbybalance()**, or **sortbydate()** functions.

```
C:\Users\power\Desktop\programing_1\FinalProjec
Do you want the data sorted by:
1.Name(enter 1)
2.Balance(enter 2)
3.Date(enter 3)
4
Invalid please enter correct number:
```

Sort by name:

Firstly, a loop is used to copy the content of "Accounts" to a new account variable "sort" and use it to sort the account and to not change it in the main struct "Accounts" because it is going to be used in another functions .then **Bubble Sorting** algorithm _is used to sort the data which compares the name of the account with the name of the following account by **strcmp()**, if the name of an account is greater than the following name, firstly: a variable called flag was initialized by 1 will turn to be 0, then the accounts are replaced using a helping account called temp. The sorting ends when the flag remains 1 till the end which will happen only if the accounts are sorted or all passes are done in the worst case.

Pseudocode representing the Bubble sort algorithm used in sort by name

```
flag = 1
pass = 1
while pass < NUM_ofaccounts and not flag
    flag = 1

    for j from 0 to NUM_ofaccounts - pass
        if strcmp(sort[j].name, sort[j + 1].name) > 0
            temp = sort[j]
            sort[j] = sort[j + 1]
            sort[j + 1] = temp
            Flag = 0
        end for
    pass = pass + 1
end while
```

```
Account Number: 9700000008
Name: Adam Mark
E-mail: ad.mark@gmail.com
Balance: 350
Mobile: 0100970008
Date Opened: October 2015
```

```
Account Number: 9700000006
Name: Daniel Graves
E-mail: dgrave@outlook.com
Balance: 450
Mobile: 01000000000
Date Opened: January 2020
```

```
Account Number: 9700000005
Name: David Roberts
E-mail: david123@gmail.com
Balance: 400.5
Mobile: 01009700005
Date Opened: October 2015
```

```
Account Number: 9700000009
Name: James Adams
E-mail: j.adams@gmail.com
Balance: 250
Mobile: 01009700009
Date Opened: May 2017
```

Sort by balance:

Same **bubble sorting** algorithm used in sorting names is used in sorting by balance but with a different comparison method by changing the balance to integer using **atoi()** function then comparing the balances.

Balance is sorted in a descending way in which the accounts with higher balance appear first. The sorting ends when the flag remains 1 till the end which will happen only if the accounts are sorted or all passes are done in the worst case.

Pseudocode representing the Bubble sort algorithm used in sort by balance

```
flag = 1
pass = 1
while pass < NUM_ofaccounts and not flag
    flag = 1
    for j from 0 to NUM_ofaccounts - pass
        if atoi(sort[j].balance) < atoi(sort[j + 1].balance)
            temp = sort[j]
            sort[j] = sort[j + 1]
            sort[j + 1] = temp
            flag = 0
    end for
    pass = pass + 1
end while
```

```
Account Number: 9700000000
Name: Michael Jones
E-mail: m.jones@gmail.com
Balance: 1000
Mobile: 01009700000
Date Opened: December 2007
```

```
Account Number: 9700000007
Name: Philipe Brian
E-mail: p.brian@outlook.com
Balance: 460
Mobile: 01009700007
Date Opened: February 2020
```

```
Account Number: 9700000006
Name: Daniel Graves
E-mail: dgrave@outlook.com
Balance: 450
Mobile: 01000000000
Date Opened: January 2020
```

```
Account Number: 9700000004
Name: Roberto Thomas
E-mail: rob.thomas@gmail.com
Balance: 400.5
Mobile: 01009700004
Date Opened: November 2015
```

Sort by date:

Also **bubble sort** algorithm is used but when comparing, it first changes the year into an integer using **atoi()** function then compares the year of the date if it is smaller than the following date year.

It replaces the accounts. otherwise if the years are equal it compares the months of the date opened as an integer and switches if the month of the first is greater. Else it stays as it is.

The sorting ends when the flag remains 1 till the end which will happen only if the accounts are sorted or all passes are done in the worst case.

Pseudocode representing the Bubble sort algorithm used in sort by date

```
flag = 1
pass = 1
while pass < NUM_ofaccounts and not flag
    flag = 1
    for j from 0 to NUM_ofaccounts - pass
        if atoi(sort[j].date.year) < atoi(sort[j + 1].date.year)
            temp = sort[j]
            sort[j] = sort[j + 1]
            sort[j + 1] = temp
            flag = 0
        else if atoi(sort[j].date.year) == atoi(sort[j + 1].date.year)
            if atoi(sort[j].date.month) < atoi(sort[j + 1].date.month)
                temp = sort[j]
                sort[j] = sort[j + 1]
                sort[j + 1] = temp
            flag = 0
        end if
    end for
    pass = pass + 1
end while
```

```
Account Number: 9700000007
Name: Philipe Brian
E-mail: p.brian@outlook.com
Balance: 460
Mobile: 01009700007
Date Opened: February 2020
```

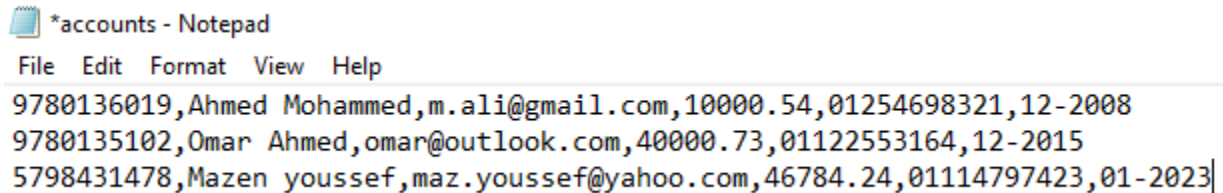
```
Account Number: 9700000006
Name: Daniel Graves
E-mail: dgrave@outlook.com
Balance: 450
Mobile: 01000000000
Date Opened: January 2020
```

```
Account Number: 9700000009
Name: James Adams
E-mail: j.adams@gmail.com
Balance: 250
Mobile: 01009700009
Date Opened: May 2017
```

```
Account Number: 9700000002
Name: Timothy Korman
E-mail: t.korman@gmail.com
Balance: 200
Mobile: 01009700002
Date Opened: December 2015
```

13 -Save

fopen() is used in write mode to overwrite the accounts.txt file, then **fprintf()** is used to print all the account fields data in the file. The process is repeated in a for loop to print the data for all accounts. At the end **fclose()** is used to close the opened file.

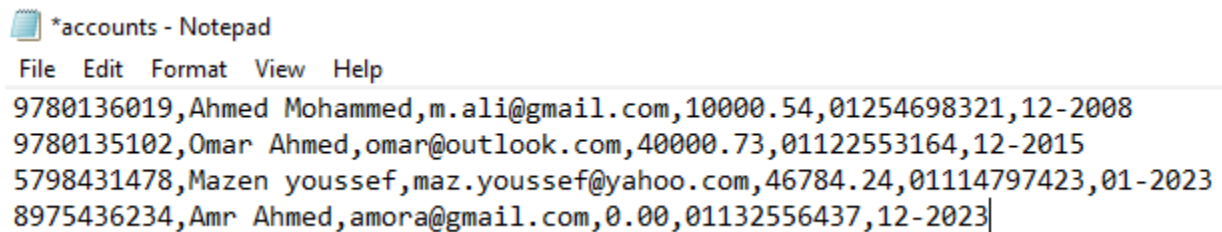


*accounts - Notepad

File Edit Format View Help

9780136019,Ahmed Mohammed,m.ali@gmail.com,10000.54,01254698321,12-2008
9780135102,Omar Ahmed,omar@outlook.com,40000.73,01122553164,12-2015
5798431478,Mazen youssef,maz.youssef@yahoo.com,46784.24,01114797423,01-2023|

account.txt before calling save()



*accounts - Notepad

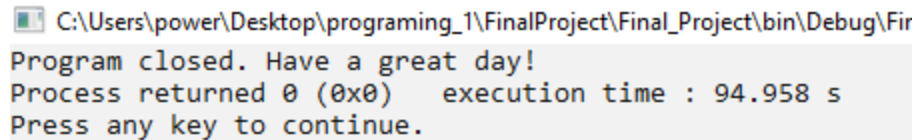
File Edit Format View Help

9780136019,Ahmed Mohammed,m.ali@gmail.com,10000.54,01254698321,12-2008
9780135102,Omar Ahmed,omar@outlook.com,40000.73,01122553164,12-2015
5798431478,Mazen youssef,maz.youssef@yahoo.com,46784.24,01114797423,01-2023
8975436234,Amr Ahmed,amora@gmail.com,0.00,01132556437,12-2023|

accounts.txt after calling save()

14 -Quit

When this function is called, it clears the terminal using **system("cls")**, then prints "Program closed. Have a great day!"



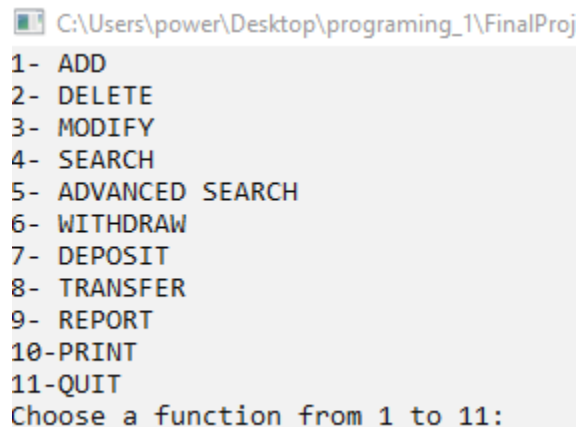
```
C:\Users\power\Desktop\programing_1\FinalProject\Final_Project\bin\Debug\Fir
Program closed. Have a great day!
Process returned 0 (0x0)   execution time : 94.958 s
Press any key to continue.
```

15- Menu

This function takes a single integer argument “P”, if “P” is equal to 1(indicating that it’s the first call of the function), the **LOAD()** will be called to load the data from the accounts.txt file into the global pointer “Accounts” of the structure account.

Furthermore, the program will print all the functions’ names asking the user to enter the number next to the function desired. After that the function the user asked for will be called.

At the end, when the called function is finished the program prints “Press any key then enter to return to the menu.”, then **system(“cls”)** is called to clear the terminal and **MENU(0)** is called again(0 indicates that it’s not the first call for menu).



```
C:\Users\power\Desktop\programing_1\FinalProj
1- ADD
2- DELETE
3- MODIFY
4- SEARCH
5- ADVANCED SEARCH
6- WITHDRAW
7- DEPOSIT
8- TRANSFER
9- REPORT
10-PRINT
11-QUIT
Choose a function from 1 to 11:
```

User Manual

A guid to help use our program

At the start of the program you will be asked to enter L to login or Q to quit

- If you entered Q, the program will end.
- If you entered L, you will be asked to enter your username and password

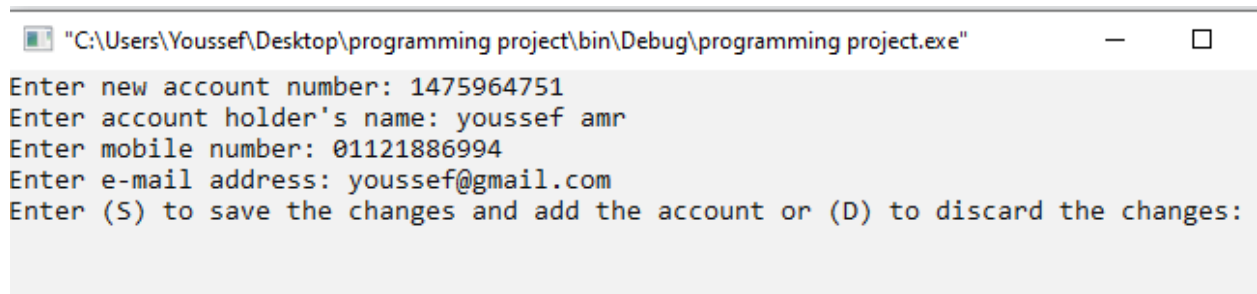
After you login successfully, you will be asked to choose a number from 1 to 11 according to the function you want to access.

If you entered 1

This will convert you to the add-account function, where you will be able to add a new account.

First when this screen appears you should start entering the data of the new account field by field

- 1- Account number: make sure you enter a 10 digit number.
- 2- Account name: make sure you enter only first and last name and don't write any numbers or special characters.
- 3- Mobile number: make sure you enter a 11 digit valid mobile number.
- 4- E-mail address: make sure your email address has the '@' symbol and a '.' after it.
- 5- Account balance: your account balance will initially be set to 0\$ (you can use the deposit function (number 7) to add a balance.



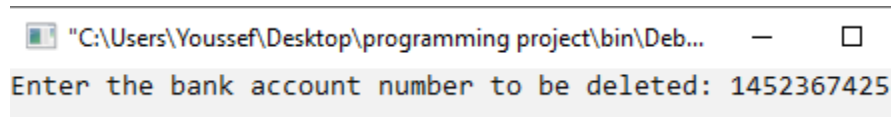
```
"C:\Users\Youssef\Desktop\programming project\bin\Debug\programming project.exe"
Enter new account number: 1475964751
Enter account holder's name: youssef amr
Enter mobile number: 01121886994
Enter e-mail address: youssef@gmail.com
Enter (S) to save the changes and add the account or (D) to discard the changes:
```

If you entered 2

This will convert you to the delete-account function, where you will be able to delete an old account.

when this screen appears you should enter the number of the account to be deleted

Make sure that the account balance is 0\$, so you would be able to delete it



If you entered 3

This will convert you to the modify function, where you will be able to modify the name or the email address or the mobile number of any account.

First you will be asked to enter the account number where the modifications will take place. Then you will be asked to enter a number from 1 to 3.

If you chose 1, you will be able to modify your account name just by entering the new name, make sure you enter only first and last name and don't write any numbers or special characters.

If you chose 2, you will be able to modify your account mobile number just by entering the new mobile number, make sure you enter a 11 digit valid mobile number.

If you chose 3, you will be able to modify your account email address just by entering the new email address, make sure your email address has the '@' symbol and a dot after it.

When you finish the modification you will be asked if you want to modify anything else, so you should Y for yes or N for no. when choosing Y the process will repeat and you should follow the same rules, but if you choose N the function will end.


```
"C:\Users\Youssef\Desktop\programming project\bin\Debug\programming project.exe"
Enter account number: 9700000001
Enter (1) to modify name
Enter (2) to modify mobile number
Enter (3) to modify email address
1
Enter the new name: ali ahmed
Enter (S) to save changes and confirm transactions or (D) to discard the changes: s
Modification was done successfully!
Do you want to modify any thing else?
Enter (Y) for yes or (N) for no: y
Enter (1) to modify name
Enter (2) to modify mobile number
Enter (3) to modify email address
2
Enter the new mobile number: 01115427894
Enter (S) to save changes and confirm transactions or (D) to discard the changes: s
Modification was done successfully!
Do you want to modify any thing else?
Enter (Y) for yes or (N) for no: n
```

If you entered 4

This will convert you to the Search function, where you will be able to access any account data, just by entering the the accounts number.

```
"C:\Users\Youssef\Desktop\programming p...
Enter account number: 9700000002
Account Number: 9700000002
Name: Timothy Korman
E-mail: t.korman@gmail.com
Balance: 200
Mobile: 01009700002
Date Opened: December 2015
```

If you entered 5

This will convert you to the advanced-search function, where you will be able to access any account data containing a certain keyword in its name, just by entering the keyword you want.

```
"C:\Users\Youssef\Desk...  -  □
Enter Keyword: Rob

Account Number: 9700000003
Name: Michael Robert
E-mail: michael@yahoo.com
Balance: 300
Mobile: 01009700003
Date Opened: November 2008

Account Number: 9700000004
Name: Roberto Thomas
E-mail: rob.thomas@gmail.com
Balance: 400.5
Mobile: 01009700004
Date Opened: November 2015

Account Number: 9700000005
Name: David Roberts
E-mail: david123@gmail.com
Balance: 400.5
Mobile: 01009700005
Date Opened: October 2015
```

If you entered 6

This will convert you to the withdraw function, where you will be able to withdraw an amount from your account balance

First you will be asked to enter the account number where the withdrawal will take place. Then you will be asked to enter the amount to be withdrawn.

- 1- make sure your initial account balance is greater than the withdrawal amount
- 2- make sure that the withdrawal amount is less than 10,000\$

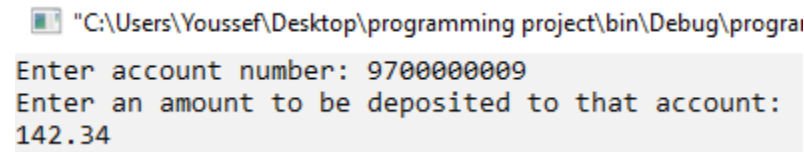
```
"C:\Users\Youssef\Desktop\programming project\bin\Debug\programm
Enter account number: 9700000009
Enter an amount to be withdrawn from that account:
100.54
```

If you entered 7

This will convert you to the deposit function, where you will be able to deposit an amount from your account balance

First you will be asked to enter the account number where the deposition will take place.

Then you will be asked to enter the amount to be deposited. Make sure that the deposition amount is less than 10,000\$.



```
"C:\Users\Youssef\Desktop\programming project\bin\Debug\program"
Enter account number: 9700000009
Enter an amount to be deposited to that account:
142.34
```

If you entered 8

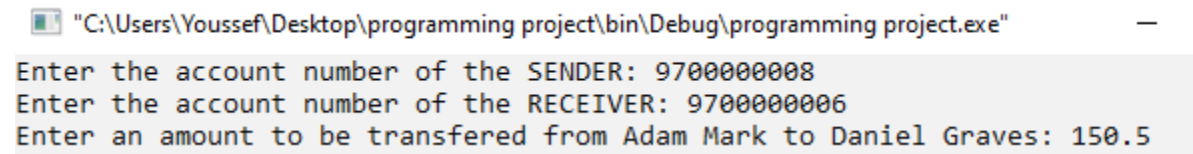
This will convert you to the transfer function, where you will be able to transfer an amount from your account balance to another account balance.

First you will be asked to enter your account number where the withdrawal will take place.

Then you will be asked to enter the account number of the receiver where the deposition will take place.

After that, you will be asked to enter the amount to be transferred.

- 1- make sure your initial account balance is greater than the transfer amount
- 2- make sure that the transfer amount is less than 10,000\$



```
"C:\Users\Youssef\Desktop\programming project\bin\Debug\programming project.exe"
Enter the account number of the SENDER: 9700000008
Enter the account number of the RECEIVER: 9700000006
Enter an amount to be transfered from Adam Mark to Daniel Graves: 150.5
```

If you entered 9

This will convert you to the report function, where you will be able to view the last 5 transaction made on a certain account, just by entering the the accounts number.

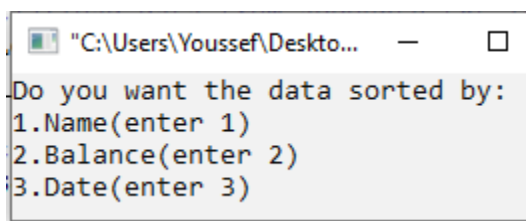
If you entered 10

This will convert you to the print function, where you will be able to view all the accounts in a sorted manner. You will be give 3 opinions for the sorting type.

If you entered 1, the accounts will be sorted by the alphabetical order of the accounts name.

If you entered 2, the accounts will be sorted by the account balance in a descending way.

If you entered 3, the accounts will be sorted according to the date when the account was made, from the newest to the oldest.



```
"C:\Users\Youssef\Desktop\...  -  □
Do you want the data sorted by:
1.Name(enter 1)
2.Balance(enter 2)
3.Date(enter 3)
```

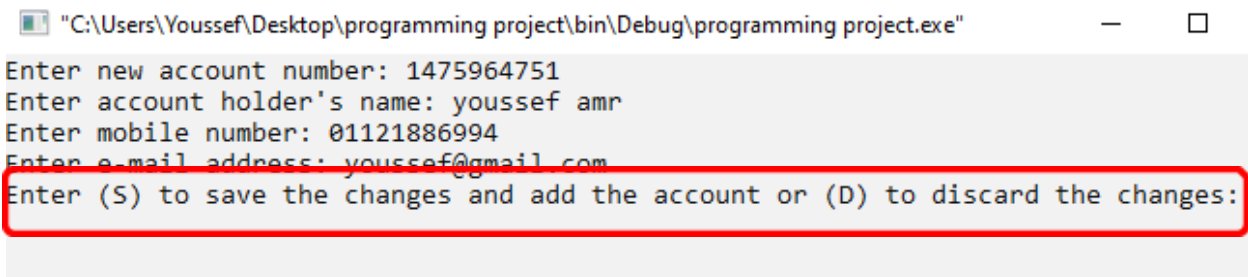
If you entered 11

You will logout and return to the initial Menu, where you will be asked to enter L to login or enter Q to quit.

At the end of each function you will be asked to enter S to save or D to discard changes.

-If you entered S, all the changes you have made will be saved and you won't be able to get your old data back.

-If you entered D, all the changes you have made will be lost.



```
"C:\Users\Youssef\Desktop\programming project\bin\Debug\programming project.exe"
Enter new account number: 1475964751
Enter account holder's name: youssef amr
Enter mobile number: 01121886994
Enter e-mail address: youssef@gmail.com
Enter (S) to save the changes and add the account or (D) to discard the changes:
```