Alexandria University
Faculty of Engineering
Specialized Scientific Programs
Computer & Communication Program

Programming (1)
Course Code: CSE126
Lecturer: Prof. Dr. Nagia Ghanem
Prof. Dr. Saleh El Shehaby

Final Project
Bank Management System

It is required to write a program that will create and maintain a Bank management system to keep track of Accounts inside the Bank.
Each Account **has an account number, name, mobile, email address, balance, date opened** (another struct of month, and year).
Your program should allow the Employees of the bank to execute the following commands:

1. **LOGIN**
   a. user is required to enter username and password to access system's functionality.
   b. you should validate user entered credentials with the username and password that are stored in a text file called "users.txt"
   Each line represents the username and password for each user.
   Example for "users.txt"

```
ahmed.mohamed 123$@1
ali.ahmed 654321
ziad.ali 123abc
```

2. **LOAD (READ FROM FILE):**
   a. This command loads the account data from a file called **"accounts.txt".** The file is a text file that is comma delimited in which each entry is found on a separate line.
   b. Each entry must contain the **account number, name, address, balance, mobile, date opened, and status**. An example of data in the file is as follows:

```
9780136019,Mohamed Ali,m.ali@gmail.com,10000.54,01254698321,12-2008, active
9780135102,Omar Ahmed,omar@outlook.com,40000.73,01122553164,12-2015, inactive
```

3. **QUERY (SEARCH):** The system should process a request by the user to look up information about a specific Account. The user must supply the Account Number, and the system should provide all data for the Account of that Number or a message indicating that the specified Account is not found.
   Data should be printed in the following format.

```
Account Number : 9780136019
Name: Mohamed Ahmed
E-mail : m.ali@gmail.com
Balance: 100000.54 $
Mobile: 01254698321
Date Opened: December 2008
Status: Active
```

4. **ADVANCED SEARCH:** The system should process a request by the user to look up information about some Accounts. The user must supply a keyword, and the system should provide all data for all accounts whose name contains that keyword or a message indicating that no matches are found.

   Example

   Enter keyword: Ahmed

   Search results:

   Account Number: 9781503204212
   Name: Ahmed Ali
   E-mail: a.ali@yahoo.com
   Balance: 56320 $
   Mobile: 01148597411
   Date Opened: December 2008
   Status: Inactive

   Account Number: 9781152304222
   Name: Mohamed Ahmed
   E-mail: m.ahmed@yahoo.com
   Balance: 100000 $
   Mobile: 01254698321
   Date Opened: February 2007
   Status: active

5. **ADD:** This function used to add a new bank account to the bank system. The system should prompt the user field by field for the data of a single account and add it to the system and update the file **"accounts.txt"**.

   Notes:
   - Date opened should be the system's current date; you don't have to enter it.
   - Account status should be active as a default setting; you don't have to enter it.
   - Make sure the account number is unique and display a message in case of a duplicate account number.

6. **DELETE:** The user should be prompted for the bank account number and the account associated with that number should be deleted from the system.

   Notes:
   - Make sure that the entered account number exists and in case of no accounts matching the entered value display an error message.
   - Deletion of Bank account is only allowed for zero balance accounts so in case of balance greater than **zero deletion of that account should be rejected**.

7. **MODIFY:** The user should be prompted for account number. The user can edit account details. The user should be prompted field by field to modify the information for that account.

   Notes:
   - You can modify only the name, mobile and email address

8. **CHANGE_STATUS:** The user can activate/deactivate the account status. The Inactive account cannot make any financial transactions.
   - **You must check if the account exist or not.**
   - You must check the account status before change it. An attention message should appear if you wish to change your current account status back to the same one.
9. **DAILYLIMIT:** Each account has a daily limit for all withdrawals made on the same day; the total withdrawal amount must not exceed the maximum withdrawal limit of 50,000$.
10. **WITHDRAW:** The user should be prompted for **an active account number**. Then the amount needs to be withdrawn from that account and after that a message displays if the transaction was successful or not.
    - A warning message should appear for inactive account showing that the process cannot completed.
    - Note that there is a maximum withdrawal limit for each transaction which is 10,000$ per transaction.
    - **You must check if the account exceeds the daily limits for withdrawal or not to complete the transaction.**
11. **DEPOSIT:** The user should be prompted for **an active account number**. Then the amount needs to be deposited to that account and after that a message displays if the transaction was successful or not.
    - Note that there is a maximum deposit limit for each transaction which is 10,000$ per transaction.
    - A warning message should appear for inactive account showing that the process cannot completed.
12. **TRANSFER**: The user can transfer money from an active bank account to another active one. The user should be prompted for account numbers of sender and receiver. The function must update the balance of both accounts.
    - A warning message should appear for inactive sender or receiver account or both showing that the process cannot completed.
13. **REPORT:** The user should be prompted for the bank account number then the function prints the last 5 transactions made on this account**.**

Notes:

- For each newly created account there should be a file named with accnumber.txt **example** : 9124123456.txt
- For each transaction [withdraw, deposit or transfer] you have to append a record for the account included in the transaction.
- Make sure you update the history of both accounts in case of transfer.

14. **PRINT(SORT):** Print the data of all accounts, in sorted order.
    You should prompt the user to choose if sort should be done based on either name or balance or Date opened.

    Note: Data should be printed the same way as in Query(Search).
    [Hint](Make 5 functions: print, sortByName, SortByDate, SortByBalance, and sortByStatus) then inside print call one of sorting functions according to user choice then print all accounts after sorting).

15. **SAVE**: Save the accounts data by writing them out to the same file in a format similar to that explained in the Load command.
    Note: after each function that modifies the data you have to ask the user if he wants to save changes and confirm transactions or discard the changes.

16. **QUIT:** Exit the program.

17. **Menu:** to allow user to choose from various functions you should provide a menu so user can select the desired option
    when the system starts initial menu will contain only LOGIN and QUIT
    - if the user logged in successfully
        - Load should be called by default
        - User can select one of the following
            - ADD
            - DELETE
            - MODIFY
            - SEARCH
            - ADVANCED SEARCH
            - CHANGE_STATUS
            - WITHDRAW
            - DEPOSIT
            - TRANSFER
            - REPORT
            - PRINT
            - QUIT
    - Otherwise
        ask the user to either go back to the menu or exit.
18. **BONUS Features:**
    - ADD a new feature of our system call **DELETEMULTIBLE; the function must allow the user to remove multiple account at once using two options accessible from a sub-menu used to select the desired option from the following:**
        - ByDate: delete every account that was created on a specific date (format YYYY-MM-DD).
        - Inactive accounts: delete every account has been inactive for more than 90 day and whose balance is zero.

Note: the function must print an appropriate message if:

1- There is no account created in the given date.
2- There is no account has an inactive status more than 90 days or its balance is equal to zero.
3- Message of a completion with the number of accounted deleted.

- Used a colorful user interface.
- Libraries are collections of code that other programs can use. They promote code reuse and modularity. **[Separate your C code into header (.h) files and source (.c) files for better organization].**

Notes:

- The program should check for errors and print appropriate messages (e.g. trying to delete an account that is **not in the file**).
- The program should check for errors and print appropriate messages (e.g. trying to make any transaction for **an exciting account**).
- **The program should validate all entered data.**
  - Validate account number (validate length of 10 digits and make sure all of them are digits).
  - Validate email address.
  - Validate all entered numbers.
  - Provide appropriate error messages for invalid operations.
  - **Your program shouldn't crash under any circumstances, even against malicious users.**
- Every task of the above tasks should be a separate function.
- You should do a menu to enable easy access for all of these functions
- Maintain a clear and user-friendly interface for the bank admins.
- Late submissions will not be accepted.
- You should work in **groups of 4 members at most.**

Cheating Policy:

- All actual programming should be an independent effort. If any kind of cheating is discovered, penalties will apply to the participating students by zero in year work marks, so delivering nothing is so much better than delivering a copy.

Deliverables:

- Report that will contain description for your work, sample runs for the program, user manual (how to use the system), and main algorithms used (search algorithm, sort algorithm).
- Source code.

☺ Good luck