CS 202 - Project 1 -- Bank Account Program

PROJECT INSTRUCTIONS

The program name shall be ***bankacct***.
It shall operate in two modes: interactive and command-line mode.
To launch the program in a **Linux** window, the user could type:

`./bankacct`	(Interactive)
`./bankacct /?`	(to display help menu)
`./bankacct` *cmd1 cmd2 ...*	(to process commands in batch mode)

In interactive mode, the program shall use menus and prompts to get information from the user.
In batch mode, it shall process the command-line arguments, and run to completion without further user interaction.

PROGRAM REQUIREMENTS

The program shall have these options:

1) creating and opening a new bank account
2) depositing funds to an existing bank account
3) withdrawing funds from an existing bank account
4) display information and balance for an existing bank account
5) transferring funds between two different existing bank accounts
6) closing out an existing bank account
7) writing a *report* file

A valid bank account number and password (both described below) shall be required for any transaction.
The program design will be based on the **structures** abstract data type (ADT).

Internally, a bank account record shall consist of the following data:

DATA	**DATA TYPE**
LAST NAME	C-String
FIRST NAME	C-String
MIDDLE INITIAL	Character
SOCIAL SECURITY NUMBER	C-String
PHONE NUMBER	C-String
ACCOUNT BALANCE	Floating Point Type
ACCOUNT NUMBER	C-String (Length 5: A,....Z,0,...,9)
ACCOUNT PASSWORD	C-String (Length 6: A,....Z,0,...,9)

RESTRICTIONS

- The main() function shall act only as an executive, controlling the flow of the program, but not implementing low-level details of it.
- No global variables shall be used.
- If any variable is needed by more than one function, it shall be declared as a local variable in main() and passed as needed.
- Validation of inputs is required. For example, the middle initial shall be restricted to A-Z, and this should be checked for. Invalid input data shall not be accepted.
- Account numbers and passwords shall not be case sensitive.
- A database entry must be complete and validated before it is added to an output file.
- The *string* class shall not be used for declaring or manipulating any data in the program.
- The user shall have a way to write a *report* file.

PROGRAM OUTPUT: BANK ACCOUNT DATABASE FILE

The program shall produce a text file that holds the bank account database.
When it is first started, the program shall inquire if a database file already exists, and what its name is.
If the file exists, it shall be opened for reading and writing.
If not, a new file of that name shall be opened for output.
Upon program exit, the output file shall be closed.

BANK ACCOUNT DATABASE FILE FORMAT

Here is the format for one record of the bank account database file.

```
Last name
First name
MI
Social security number
Phone number
Account Balance
Account Number
Account Password
```

BANK ACCOUNT DATABASE FILE

The DB file shall have the exact format exhibited below, with bank account records delimited by blank lines.
The following example bank account database file contains three records.

```
Richards
Steven
A
123-89-4321
(775)332-4581
76809.21
a123b
a23b42

Smith
Shelly
B
323-77-2134
(775)785-1291
126812.33
c123a
z52c42

Kim
Justin
M
782-23-5512
(702)132-1351
29122.35
f123c
u2xc90
```

PROGRAM OUTPUT: BANK ACCOUNT DATABASE REPORT FILE

When the user chooses to write a *report*, a *report* file shall be written.
The default report file name shall be: ***BankAcct.Rpt***
The program shall give the user the choice of picking a different name for the report file.
If the user chooses to do so, the user shall be prompted to enter the alternate name of the report file.
The report file shall include the same information as the bank account database output file, but it shall be formatted for human viewing, as a table.
The `#include` file `<iomanip>` shall be used for this purpose.

BANK ACCOUNT DATABASE REPORT FILE

The following example bank account database report file contains the same three records, but is formatted and, therefore, easier for a human to read.

```
-------    ----        -----     --   -----------   -------------   ---------
Account    Last        First     MI   SS            Phone           Account
Number     Name        Name           Number        Number          Balance
-------    ----        -----     --   -----------   -------------   ---------
a123b      Richards    Steven    A    123-89-4321   (775)332-4581    76809.21
c123a      Smith       Shelly    B    323-77-2134   (775)785-1291   126812.33
f123c      Kim         Justin    M    782-23-5512   (702)132-1351    29122.35
```

COMMAND-LINE MODE

The command lines:

`./bankacct /?` shall cause a help screen to appear, describing how to use the program.
`./bankacct` *cmd1 cmd2 ...* shall cause commands to be executed in batch mode.

Here are the valid command line parameters:

`/?`	`/Nphone_num`	`/Wwithdraw_amt`	
`/Llast_name`	`/Aacct_num`	`/Ttransfer_amt`	
`/Ffirst_name`	`/Ppassword`	`/Odb_file`	
`/Minitial`	`/Cnew_password`	`/Rrpt_file`	
`/Sssn`	`/Ddeposit_amt`	`/I`	(displays account information)

COMMAND LINE PARAMETER DESCRIPTIONS

The meanings of the command line parameters are described below.
When multiple command line parameters appear on the same line, they can appear in any order.
The independence of order of the parameters is demonstrated in the "equivalent examples" below.
Note that all examples except the last must be accompanied by the /O, /A and /P command line parameters.
And all examples require reading and writing the database file.

1. `bankacct /Odb_file /Aacct_num /Ppassword /I`
   Read an input file that holds the bank account database, the name of which is "db_file."
   Display all information for account with provided "account number" and "account password."
   Example: `bankacct /OBankDB.txt /Aa123b /Pa23b42 /I`

2. `bankacct /Odb_file /Aacct_num /Ppassword /Llast_name`
   Change "last name" for account with provided "account number" and "account password."
   Example: `bankacct /OBankDB.txt /Aa123b /Pa23b42 /LRichards`

3. `bankacct /Odb_file /Aacct_num /Ppassword /Ffirst_name`
   Change "first name" for account with provided "account number" and "account password."
   Example: `bankacct /OBankDB.txt /Aa123b /Pa23b42 /FSteven`

4. `bankacct /Odb_file /Aacct_num /Ppassword /Minitial`
   Change "middle initial" for account with provided "account number" and "account password."
   Example: `bankacct /OBankDB.txt /Aa123b /Pa23b42 /MA`

5. `bankacct /Odb_file /Aacct_num /Ppassword /Sssn`
   Change "social security number" for account with provided "account number" and "account password."
   Equivalent examples:
   ```
   bankacct /OBankDB.txt /Aa123b /Pa23b42 /S123-89-4321
   bankacct /Aa123b /Pa23b42 /S123-89-4321 /OBankDB.txt
   ```

6. `bankacct /Odb_file /Aacct_num /Ppassword /Nphone_num`
   Change "phone number" for account with provided "account number" and "account password."
   Equivalent examples:
   ```
   bankacct /OBankDB.txt /Aa123b /Pa23b42 /N(775)332-4581
   bankacct /Aa123b /Pa23b42 /OBankDB.txt /N(775)332-4581
   ```

7. `bankacct /Odb_file /Aacct_num /Ppassword /Ddeposit_amt`
   `bankacct /Odb_file /Aacct_num /Ppassword /Wwithdraw_amt`

   Deposit or withdraw "amount" for account with provided "account number" and "account password."

   Examples: `bankacct /OBankDB.txt /Aa123b /Pa23b42 /D12309.20`
   `bankacct /OBankDB.txt /Aa123b /Pa23b42 /W55809.02`

8. `bankacct /Odb_file /Aacct_num /Ppassword /Aacct_num /Ppassword /Ttrans_amt`
   Transfer "amount" from the first "account number" and "account password" pair to the second "account number" and "account password" pair. Two different accounts must be used.

   Example: `bankacct /OBankDB.txt /Aa123b /Pa23b42 /Ac123a /Pz52c42 /T76809.20`

   Transfer from "account number" a123b with "password" a23b42 to "account number" c123a with "password" z52c42 an amount of $76809.20. If insufficient funds, do not transfer anything.

9. `bankacct /Odb_file /Aacct_num /Ppassword /Cnewpassword`
   Change "password" for account with provided "account number" and "account password."

   Equivalent examples:
   ```
   bankacct /OBankDB.txt /Aa123b /Pa23b42 /CZZSTOP
   bankacct /Aa123b /Pa23b42 /CZZSTOP /OBankDB.txt
   ```

   Change the password of "account number" a123b with "password" a23b42 to have the new password ZZSTOP.

10. `bankacct /Odb_file /Rrpt_file`
    Produce a "report" file of the bank account database, put the report in a file named "rpt_file."

    Equivalent examples:
    ```
    bankacct /OBankDB.txt /RBankRpt.txt
    bankacct /RBankRpt.txt /OBankDB.txt
    ```

CODING CONVENTION

- Each file shall have a file header.
- Each function shall have a function header.
- Your code shall be properly indented and commented.

GENERAL PROJECT REQUIREMENTS

Programs submitted as part of a project must have an executable built on the Linux server.
Programs submitted as part of a project must satisfy the requirements of the project.
The student must present the project to the class, discussing the project's requirements and how the program satisfies them, giving sample runs for the program, walking through the source codes for the program, discussing the project's makefile, discussing the program's shortcomings and known bugs.

SUBMISSION REQUIREMENTS

Submit a "tgz" file named ***proj1.tgz*** that contains the entire project: executable, source files, makefile, and data files. Make sure all your files are in your: **Proj1** directory.

`cd ..`	to move up one level in your directory structure.
`tar -czvf proj1.tgz Proj1`	to create a Linux archive ZIP file.
`tar -tvf proj1.tgz`	to see an index of what is in the "tgz" file.

Use FileZilla to copy your `".tgz"` file from the Linux server over to your Windows machine.
Upload the "tgz" file to Canvas.

Submission verification is possible because you are allowed to download files from Canvas.
You should test your submission by performing the following procedure:

1. Download your "tgz" file from the Canvas drop box.
2. Unzip your "tgz" file (extract its contents).
3. Transfer the file(s) to the server.
4. Be sure to do this in a different "test" directory, so you don't overwrite previous work.
5. Run the resulting program.