Project Five
Check Register with Data Files

Review and run the code in the attached checkRegister.cpp file. Use the data in *checkIn.dat* for user input.

Change the program so it uses data from a file rather than user entries to process a customer's bank transactions. The file with the beginning balance and the customer transactions is named *checkIn.dat*. Open this file so you can see how the data is set up. The first field is the beginning balance followed by a series of records with transaction codes and amounts.

Make the following changes:

- Change the code so that the program reads the balance and transactions from checkIn.dat using the **provided** functions. The code should work no matter how many records are in the file. Use a structure to define the record. The record structure has two members: code and amount. Eliminate the user prompts currently in the code, but still list the transactions as displayed by checkRegister.cpp.

  Keep the *cout* statements in the program as is because the project should still list the transactions as well as the totals at the end.

  Change the getBegBal function to read the beginning balance from the file. Add a call to displayBal after reading the beginning balance from the file so the beginning balance is displayed. You need to do this because you deleted the prompt for beginning balance. The getData function should be changed to get the data from the file instead of the user, and return a structure.

  Write the final balance to an output file. Name this file *checkOut.dat*

- Display the following dollar totals at the end of the report.

  - Credits (additions to the account – deposits)
  - Debits (deductions from the account – checks and ATM withdrawals)
  - Service charges (ATM charges and negative balance charges)

  Display the number of transactions in the file (this is a counter).

  Use a structure definition for the totals including the transaction counter. Update the total structure members in main. Add a function to **display** the totals, sending the totals structure.

Purpose of this Project

Develop C++ program with the following new features:

- Data files (Chapter Six plus the Blackboard notes)
  Check the last sample in the *Chapter Six Notes* for file processing syntax.

  The while loop will look like this to process a file:

  ```
  while(!inFile.eof())
  {
          trans = GetData(inFile);
          …..
  ```
  inFile is the name of the check file and trans is the name of the structure variable that contains the code and amount members.

- Structures (*Chapter Ten – 10.1 Structures* plus the Blackboard notes)

  The sample *payfileFunction.cpp* file I attached with the project instructions is an excellent guide to adding structures and functions to the project.

Advice from the Instructor:

- Review the data in the file and set up a spreadsheet to determine what the correct results should be so you can verify the results of your testing.

- Try to use the root directory of your disk to store the files to avoid problems with accessing the files. When you open a file, note the doubles slashes in the following example so that the compiler doesn't interpret the backslash as an escape character.

  ```
  inFile.open("c:\\checkIn.dat");
  ```

  For Windows 10, you may need to use:

  ```
  inFile.open("c:\\folder\\checkIn.dat");
  ```
  *folder is any folder name

- For mac users: Use the forward slash to separate directories instead of the back slash. If the file is saved on your desktop then you would enter something like

  ```
  inFile.open ("/Users/Name/Desktop/checkIn.dat");
  ```

  The directories, name, and file name are case sensitive.

- Be sure to check for a file open failure and display a message.