CPS 171 Machine Problem 7 Due 04/22/2002

You are going to modify your ***Account*** class that you created in **MP6**. Also you are going to write a new program to test that modified class and simulate a day at the bank.

The ***Account*** class should now have 3 data members, a string **my_accNum**, a string **my_password**, and a double **my_balance**.

The Account class will have now only the default constructor. It will initialize the string data members **my_accNum** and **my_password** to an empty string and **my_balance** to zero.

Add a new method to the ***Account*** class called ***setAccount***(). This method will take 3 parameters, **account_Number, password** and **amount**. Assign the parameters to the corresponding class data members.

The class will also have the following four methods:

make_deposit(string, double)
will remain the same as before in MP6.

make_withdraw(string, double)
will remain the same as before in MP6.

double get_balance()
will return the balance without checking for the password.

string get_account()
will return the account without checking for the password.

Remove all other methods from your ***Account*** class.

Create a new program ***MP7Array*** that will create an array of ***Account*** class objects. The maximum number of elements in the array will be 10 but the data file may or may not contain that many records. The program will read account numbers, passwords, and starting amounts from a file "mp7master.dat", echoing the data as it is read. Every time a record is read from the file, the program will use the **setAccount()** method to assign the account number, the password and the amount to that object in the array. Keep a counter called **number_of_accounts** to count how many accounts were created and print out its value when the end of file is reached.

The program will then read the day's transactions at the bank from another file "mp7trans.dat". Every record in this file will have account number, password, amount, and transaction code (W or D). For each transaction you must find the correct account in the array and update the balance. Keep doing this until the end of the file is reached.

The code can be either **W** or **w** for withdrawal or **D** or **d** for deposit. If the code is **W**, the amount should be withdrawn from the balance. If the code is **D**, the amount will be added to the balance.

Note: Any bad data such as invalid code, invalid amount or invalid password should be reported as an error message. While debugging, you might want to print out information about each transaction, though this is not required in the final output.

At the end of the day at the bank, sort the array of the objects in ascending order by balance or by account number (whichever the user wishes), and then display all the accounts and their balances.

You should use at least 3 functions in your program (you may use more if you wish):

int searchArray(account [], int, string)

The function will have three parameters, the array of objects, the size of the array, and the account you are searching for. Then it will return either the index of the account if it was found or –1 if it was not found.

void sortArray(account [], int)

The function will have two parameters, the array of objects, and the size of the array. Then it will ask the user if the sort will be done by account or by balance. It will then sort the array by that choice.
The algorithm on Page 757 of the textbook may be helpful here but it will need some changes for this application.

void printArray(account [], int)

The function will have two parameters, the array of objects, and the number of accounts in the array. Then it will print the objects' account numbers and their balance. Also it will print the total number of the accounts.

You are to use the input files available on the college network at **t:\class\cps\cps171\datafiles\mp7master.dat** and **mp7trans.dat**. Copy these files to the same subdirectory as your program.

You should turn in a printed copy of your program and two sets of output – the first sorted by account numbers and the second sorted by the balances. The program should not be changed between the two sets.

If you were not able to get MP6 working correctly, a sample solution will be provided on **Monday April 15 at 9am**. It will be on the network at **t:\class\cps\cps171\mp6sol**
We recommend that you try to fix your own solution before using this.

A sample input from the input file "mp7master.dat"
1234 funstuff 500.50
2345 welcome 150.00

A sample record from the input file "mp7trans.dat"
2345 welcome 100.00 W
2222 welcome 100.00 W
1234 welcome 75.00 X
2345 funstuff 90.00 W
1234 welcome 100.00 D

A sample output would be:

Original Data

Account Number	Password	Balance
1234	funstuff	500.50
2345	welcome	150.00

There are 2 accounts

Processing transactions
2222=> invalid account. 2222
1234=> invalid code. X
2345=> invalid password. Funstuff

Do you like to sort [1] by account or [2] by amount: 1

Account	amount
1234	600.50
2345	50.00

There are 2 accounts