CH12 #7: Write a checkbook balancing program.

The program will read in, from the console, the following for all checks that were not cashed as of the last time you balanced your checkbook:
the number of each check (int),
the amount of the check (double),
and whether or not it has been cashed (1 or 0, boolean in the array).
Use an array with the class as the type. The class should be a class for a check.

There should be three member variables to record the check number, the check amount, and whether or not the check was cashed.

The class for a check will have a member variable of type Money (as defined on page 662 in the book; Display 11.9) to record the check amount.

So, you will have a class used within a class. The class for a check should have accessor and mutator functions as well as constructors and functions for both input and output of a check.

In addition to the checks, the program also reads all the deposits (from the console; cin), the old and the new account balance (read this in from the user at the console; cin).

You may want another array to hold the deposits. The new account balance should be the old balance plus all deposits, minus all checks that have been cashed.

The program outputs the total of the checks cashed, the total of the deposits, what the new balance should be, and how much this figure differs from what the bank says the new balance is.

It also outputs two lists of checks: the checks cashed since the last time you balanced your checkbook and the checks still not cashed.

[edit: if you can, Display both lists of checks in sorted order from lowest to highest check number.]

DISPLAY 11.9 Program Using an Array of `Money` Objects *(part 1 of 2)*

```
1   //This is the definition for the class Money.
2   //Values of this type are amounts of money in U.S. currency.
3   #include <iostream>
4   using namespace std;
5   class Money
6   {
7   public:
8   friend Money operator +(const Money& amount1, const Money& amount2);
9   //Returns the sum of the values of amount1 and amount2.
10  friend Money operator -(const Money& amount1, const Money& amount2);
11  //Returns amount1 minus amount2.
12  friend Money operator -(const Money& amount);
13  //Returns the negative of the value of amount.
14  friend bool operator ==(const Money& amount1, const Money& amount2);
15  //Returns true if amount1 and amount2 have the same value; false otherwise.
16  friend bool operator <(const Money& amount1, const Money& amount2);
17  //Returns true if amount1 is less than amount2; false otherwise.
18  Money(long dollars, int cents);
19  //Initializes the object so its value represents an amount with
20  //the dollars and cents given by the arguments. If the amount
21  //is negative, then both dollars and cents should be negative.
22  Money(long dollars);
23  //Initializes the object so its value represents $dollars.00.
24  Money( );
25  //Initializes the object so its value represents $0.00.
26  double get_value( ) const;
27  //Returns the amount of money recorded in the data portion of the calling
28  //object.
29  friend istream& operator >>(istream& ins, Money& amount);
30  //Overloads the >> operator so it can be used to input values of type
31  //Money. Notation for inputting negative amounts is as in - $100.00.
32  //Precondition: If ins is a file input stream, then ins has already been
33  //connected to a file.
34
35  friend ostream& operator <<(ostream& outs, const Money& amount);
36  //Overloads the << operator so it can be used to output values of type
37  //Money. Precedes each output value of type Money with a dollar sign.
38  //Precondition: If outs is a file output stream, then outs has already been
39  //connected to a file.
40  private:
41      long all_cents;
42  };
43
```

(continued)

DISPLAY 11.9 Program Using an Array of `Money` Objects *(part 2 of 2)*

<The definitions of the member functions and the overloaded operators goes here.>

```
44  //Reads in 5 amounts of money and shows how much each
45  //amount differs from the largest amount.
46  int main( )
47  {
48      Money amount[5], max;
49      int i;
50      cout << "Enter 5 amounts of money:\n";
51      cin >> amount[0];
52      max = amount[0];
53      for (i = 1; i < 5; i++)
54      {
55          cin >> amount[i];
56          if (max < amount[i])
57              max = amount[i];
58          //max is the largest of amount[0], . . ., amount[i].
59      }
60      Money difference[5];
61      for (i = 0; i < 5; i++)
62          difference[i] = max - amount[i];
63      cout << "The highest amount is " << max << endl;
64      cout << "The amounts and their\n"
65           << "differences from the largest are:\n";
66      for (i = 0; i < 5; i++)
67      {
68          cout << amount[i] << " off by "
69               << difference[i] << endl;
70      }
71      return 0;
72  }
```

Sample Dialogue

```
Enter 5 amounts of money:
$5.00 $10.00 $19.99 $20.00 $12.79
The highest amount is $20.00
The amounts and their
differences from the largest are:
$5.00 off by $15.00
$10.00 off by $10.00
$19.99 off by $0.01
$20.00 off by $0.00
$12.79 off by $7.21
```