```
Script started on 2024-11-07 13:58:04-06:00 [TERM="xterm-256color" TTY="/dev/pts/6'
e_prevost@ares:~/Portfolio_2/Project 1-new$ pwd
/home/students/e_prevost/Portfolio_2/Project 1-new
e_prevost@ares:~/Portfolio_2/Project 1-new$ cat checkbook.info
****************

Robert Prevost

CSC 122 W01



Checkbook Project



Takes in Check information and tracks checks and deposites and calculates

balances and shows cashed and  uncashed checks



Base Level: Level 4.5

Bonus for input: +1 Level

Total Level: Level 5.5

****************e_prevost@ares:~/Portfolio_2/Project 1-new$ show-code Check.h
```

Check.h:

```
 1  #ifndef CHECK_H
 2  #define CHECK_H
 3
 4  #include "Money.h"
 5
 6  class Check {
 7  private:
 8      int checkNumber;
 9      Money amount;
10      bool hasBeenCashed;
11
12  public:
13      Check() {
14          checkNumber = 0;
15          hasBeenCashed = false;
16      }
17
18      void setCheckNumber(int number) {
19          checkNumber = number;
20      }
21
22      void setAmount(const Money& newAmount) {
23          amount = newAmount;
24      }
25
26      void setCashed(bool cashed) {
27          hasBeenCashed = cashed;
28      }
29
30      int getCheckNumber() const {
31          return checkNumber;
32      }
33
34      Money getAmount() const {
35          return amount;
36      }
37
38      bool isCashed() const {
39          return hasBeenCashed;
40      }
41  };
42
43  #endif
```

```
e_prevost@ares:~/Portfolio_2/Project 1-new$ show-code Money.h
```

Money.h:

```
 1  // This is the HEADER FILE money.h.  This is the INTERFACE for the class
 2  // Money.  Values of this type are amounts of money in U.S. currency.
 3
 4  #ifndef MONEY_H
 5  #define MONEY_H
 6
 7  #include <iostream>
 8
 9  class Money
10  {
11      long all_cents;              // monetary value stored as pennies
12
13  public:
14
15      // Initializes the object to $0.00.
16      Money(void);
17
18      // Initializes the object to dollars*100 cents.
19      Money(long dollars);
20
21      // Initializes the object to dollars*100 + cents.
22      Money(long dollars, short cents);
23
24      // Postcondition:  return value is sum of calling object and amount.
25      //                 neither amount nor calling object are changed.
```

```
    26      Money add(const Money & amount) const;
    27
    28      // Postcondition:  return value is difference of calling object
    29      //andamount.
    30      //                neither amount nor calling object are changed.
    31      Money subtract(const Money & amount) const;
    32
    33      // Postcondition:  return value is arithmetic negation of calling
    34      //object.
    35      //                calling object is not changed.
    36      Money negate(void) const;
    37
    38      // Returns true if the calling object equals the amount, false
    39      //otherwise.
    40      bool equals(const Money & amount) const;
    41
    42      // Returns true if the calling object is less than the amount,
    43      // false otherwise.
    44      bool less(const Money & amount) const;
    45
    46      // Postcondition:  calling object's value is read from the stream
    47      //                 in normal U.S. format:  $ddd.cc.
    48      void input(std::istream & ins);
    49
    50      // Postcondition:  calling object's value is printed on the stream
    51      //                 in normal U.S. format:  $ddd.cc.  (calling object
    52      //                 is not changed)
    53      void output(std::ostream & outs) const;
    54
    55      // Returns amount of money in decimal format.
    56      double get_value(void) const;
    57  };
    58
    59  #endif
```

e_prevost@ares:~/Portfolio_2/Project 1-new$ show-code Money.cpp

Money.cpp:

```
     1  #include "Money.h"
     2  #include <cstdlib>
     3  #include <iomanip>
     4
     5  Money::Money(void) {
     6      all_cents = 0;
     7  }
     8
     9  Money::Money(long dollars) {
    10      all_cents = dollars * 100;
    11  }
    12
    13  Money::Money(long dollars, short cents) {
    14      all_cents = dollars * 100 + cents;
    15  }
    16
    17  Money Money::add(const Money & amount) const {
    18      Money temp;
    19      temp.all_cents = all_cents + amount.all_cents;
    20      return temp;
    21  }
    22
    23  Money Money::subtract(const Money & amount) const {
    24      Money temp;
    25      temp.all_cents = all_cents - amount.all_cents;
    26      return temp;
    27  }
    28
    29  Money Money::negate(void) const {
    30      Money temp;
    31      temp.all_cents = -all_cents;
    32      return temp;
    33  }
    34
    35  bool Money::equals(const Money & amount) const {
    36      return all_cents == amount.all_cents;
    37  }
    38
    39  bool Money::less(const Money & amount) const {
    40      return all_cents < amount.all_cents;
    41  }
    42
    43  void Money::input(std::istream & ins) {
    44      char dollar_sign;
    45      long dollars;
    46      char decimal_point;
    47      short cents;
    48      ins >> dollar_sign >> dollars >> decimal_point >> cents;
    49      all_cents = dollars * 100 + cents;
    50  }
    51
    52  void Money::output(std::ostream & outs) const {
    53      long absolute_cents = abs(all_cents);
    54      long dollars = absolute_cents / 100;
    55      short cents = absolute_cents % 100;
    56      if (all_cents < 0)
    57          outs << "-";
    58      outs << "$" << dollars << '.' << std::setw(2) << std::setfill('0') << c
    59  }
    60
    61  double Money::get_value(void) const {
    62      return all_cents / 100.0;
    63  }
```

e_prevost@ares:~/Portfolio_2/Project 1-new$ show-code checkbook.cpp

checkbook.cpp:

```cpp
#include <iostream>
#include "Check.h"
#include "Money.h"
using namespace std;

int main() {
    Money oldBalance, newBalance;
    int numChecks;
    int numDeposits;

    Check* checks; //dynamic arrays
    Money* deposits;

    cout << "Enter your previous balance ($dd.cc format): ";
    oldBalance.input(cin);

    cout << "How many checks do you have? ";
    cin >> numChecks;
    checks = new Check[numChecks];

    for (int i = 0; i < numChecks; i++) {
        int checkNum;
        char isCashed;
        Money amount;

        cout << "\nCheck #" << (i + 1) << endl;
        cout << "Enter check number: ";
        cin >> checkNum;
        cout << "Enter check amount ($dd.cc format): ";
        amount.input(cin);
        cout << "Has this check been cashed? (y/n): ";
        cin >> isCashed;

        checks[i].setCheckNumber(checkNum);
        checks[i].setAmount(amount);
        checks[i].setCashed(isCashed == 'y' || isCashed == 'Y');
    }

    cout << "\nHow many deposits do you have? ";
    cin >> numDeposits;
    deposits = new Money[numDeposits];

    for (int i = 0; i < numDeposits; i++) {
        cout << "Enter deposit #" << (i + 1) << " amount ($dd.cc format):"
        deposits[i].input(cin);
    }

    cout << "\nEnter your new balance from bank ($dd.cc format): ";
    newBalance.input(cin);

    Money totalDeposits;
    Money totalCashedChecks;

    for (int i = 0; i < numDeposits; i++) {
        totalDeposits = totalDeposits.add(deposits[i]);
    }

    for (int i = 0; i < numChecks; i++) {
        if (checks[i].isCashed()) {
            totalCashedChecks = totalCashedChecks.add(checks[i].getAmount
            ());
        }
    }

    Money expectedBalance = oldBalance.add(totalDeposits).subtract
    (totalCashedChecks);
    Money difference = newBalance.subtract(expectedBalance);

    cout << "\n----- RESULTS -----\n";
    cout << "Total Deposits: ";
    totalDeposits.output(cout);
    cout << endl;

    cout << "Total Cashed Checks: ";
    totalCashedChecks.output(cout);
    cout << endl;

    cout << "Expected Balance: ";
    expectedBalance.output(cout);
    cout << endl;

    cout << "Bank's Balance: ";
    newBalance.output(cout);
    cout << endl;

    cout << "Difference: ";
    difference.output(cout);
    cout << endl;

    // Print cashed checks
    cout << "\nCashed Checks:\n";
    for (int i = 0; i < numChecks; i++) {
        if (checks[i].isCashed()) {
            cout << "Check #" << checks[i].getCheckNumber() << ": ";
            checks[i].getAmount().output(cout);
            cout << endl;
        }
    }

    // Print uncashed checks
    cout << "\nUncashed Checks:\n";
    for (int i = 0; i < numChecks; i++) {
        if (!checks[i].isCashed()) {
            cout << "Check #" << checks[i].getCheckNumber() << ": ";
            checks[i].getAmount().output(cout);

            cout << endl;
```

```
   108          }
   109      }
   110
   111      // Clean up dynamic arrays
   112      delete[] checks;
   113      delete[] deposits;
   114      return 0;
   115  }
e_prevost@ares:~/Portfolio_2/Project 1-new$ CPP checkbook Money Check
Money.cpp...
checkbook.cpp***
Money.cpp: In constructor
'Money::Money()':
Money.cpp:5:1: warning:
'Money::all_cents' should be initialized in
the member initialization list [-Weffc++]
    5 | Money::Money(void) {
      | ^~~~~
Money.cpp: In constructor 'Money::Money(long
int)':
Money.cpp:9:1: warning:
'Money::all_cents' should be initialized in
the member initialization list [-Weffc++]
    9 | Money::Money(long dollars) {
      | ^~~~~
Money.cpp: In constructor 'Money::Money(long
int, short int)':
Money.cpp:13:1: warning:
'Money::all_cents' should be initialized in
the member initialization list [-Weffc++]
   13 | Money::Money(long dollars, short cents) {
      | ^~~~~
Money.cpp: In member function 'void
Money::output(std::ostream&) const':
Money.cpp:53:31: warning: conversion
from 'long int' to 'int' may
change value [-Wconversion]
   53 |     long absolute_cents = abs(all_cents);
      |                           ^~~~~~~~~
Money.cpp:55:34: warning: conversion
from 'long int' to 'short int'
may change value [-Wconversion]
   55 |     short cents = absolute_cents % 100;
      |                   ~~~~~~~~~~~~~~~^~~~~
Money.cpp: In member function 'double
Money::get_value() const':
Money.cpp:62:12: warning: conversion
from 'long int' to 'double' may
change value [-Wconversion]
   62 |     return all_cents / 100.0;
      |            ^~~~~~~~
In file included from checkbook.cpp:2:
Check.h: In constructor
'Check::Check()':
Check.h:13:5: warning:
'Check::checkNumber' should be initialized
in the member initialization list [-Weffc++]
   13 |     Check() {
      |     ^~~~~
Check.h:13:5: warning:
'Check::amount' should be initialized in the
member initialization list [-Weffc++]
Check.h:13:5: warning:
'Check::hasBeenCashed' should be initialized
in the member initialization list [-Weffc++]


e_prevost@ares:~/Portfolio_2/Project 1-new$ ./checkbook.out
Enter your previous balance ($dd.cc format):
$10.00
How many checks do you have? 2

Check #1
Enter check number: 20
Enter check amount ($dd.cc format): $40.00
Has this check been cashed? (y/n): y

Check #2
Enter check number: 100
Enter check amount ($dd.cc format): $100.20
Has this check been cashed? (y/n): n

How many deposits do you have? 2
Enter deposit #1 amount ($dd.cc format):
$100.50
Enter deposit #2 amount ($dd.cc format):$20.00

Enter your new balance from bank ($dd.cc format):

$300.00

----- RESULTS -----
Total Deposits: $120.50
Total Cashed Checks: $40.00
Expected Balance: $90.50
Bank's Balance: $300.00
Difference: $209.50

Cashed Checks:
Check #20: $40.00

Uncashed Checks:
Check #100: $100.20
e_prevost@ares:~/Portfolio_2/Project 1-new$ cat checkbook.tpq
cat: checkbook.tpq: No such file or directory
e_prevost@ares:~/Portfolio_2/Project 1-new$ exit
exit
```

```
Script done on 2024-11-07 14:01:42-06:00 [COMMAND_EXIT_CODE="1"]
```