```
Script started on Tue 31 Oct 2017 03:34:28 AM CDT
\033]0;p_stach@mars:~/Balance\007[p_stach@mars Balance]$ pwd
/home/students/p_stach/Balance
\033]0;p_stach@mars:~/Balance\007[p_stach@mars Balance]$ cat main.info
--------------------------------------------------------------------------------
Name: Patrick Stach

Class: CSC122-001

Program: Project - "Balance in all things"

Levels Attempted:
-This assignment is (Level 4.5).
Add (Level 2) to allow the user to have their checks' information in a file
or to type it in at the keyboard. You decide on the file's data format.
4.5+2 = Level 6.5

Program Description:
This program assists with balancing checkbooks when given a starting balance,
and checks/deposits asinputs. Checks can either be entered by keyboard
or imported with a file. The user chooses how many checks and how many deposits
will be inputted. Given a reported balance from the bank, the program will
automatically calculate the difference between the reported balance and the
calculated balance. The program will also list cashed checks and
uncashed checks by check # from smallest to greatest.
\033]0;p_stach@mars:~/Balance\007[p_stach@mars Balance]$ cat main.cpp
#include "Money.h"
#include "Check.h"
#include <string>
#include <iostream>
#include <climits>
#include <fstream>

using namespace std;

// Function to swap two checks
inline void Swap(Check &x, Check &y)
{
        Check tmp = x;
        x = y;
        y = tmp;
}

const string divider = string(60, '-') + '\n';

int main()
{

        cout << "Welcome to the Checkbook Balancing Program\n";
        cout << "Cash amounts are entered in the form [$ddd.cc]\n";

        cout << divider << "Enter last month's balance: ";
        bool has_space = true; // indicator if enough space was made for array
        Money last_balance;
        last_balance.input(cin);
        cout << divider << "Enter the balance reported by bank: ";
        Money bank_balance;
        bank_balance.input(cin);
        Check *pcheck = NULL; // creates pointer for check array
        long i, num_checks;
        cout << divider << "How many checks will you enter? ";
        cin >> num_checks;
        while (num_checks < 1)
        {
                cout << "Invalid number.\nHow many checks will you enter? ";
                cin >> num_checks;
        }
        pcheck = new Check[num_checks]; //allocates memore for check array
        Money check_total;
        if (pcheck != NULL) //could allocate space
        {
                char choice;
                cout << divider << "Input from file? y/n: ";
                cin >> choice;
                while (choice != 'y'  && choice != 'n')
                {
                        cout << "Invalid option.\nInput from file? y/n : ";
                        cin >> choice;
                }

                //***** Inputting from File *****//
                if (choice == 'y')
                {
                        //File Stream
                        ifstream input;
                        string filename;

                        cin.clear();
                        cin.ignore(INT_MAX, '\n');
                        cout << "Enter file name: ";

                        //Entering filename, checking if it exists
                        getline(cin, filename);
                        input.open(filename.c_str());
                        while (!input)
                        {
                                input.close();
                                input.clear();

                                cout << "File does not exist.\nEnter file name: ";
                                getline(cin, filename);
                                input.open(filename.c_str());
                        }
                        cout << "'" << filename << "' selected as input file." << endl
;
                        // inputs checks into array
                        for (i = 0; i < num_checks; i++)
                        {
                                pcheck[i].input(input);
                        }
                        // sums total for cashed checks
                        for (i = 0; i < num_checks; i++)
                        {
                                if (pcheck[i].get_cashed())
                                {
                                        check_total = (pcheck[i].get_amount()).add(che
ck_total);
                                }
                        }
                        cout << divider << "Total from cashed checks:  ";
                        check_total.output(cout);
                        cout << endl;
                        input.close();
                }
                else
                        //***** Inputting from Keyboard *****//
                {
                        cout << "Enter data in the format (without braces):\n"
```

```
                                "[Check Number] [Check Amount] [Cashed? y/n]\n";
                        //input checks into array
                        for (i = 0; i < num_checks; i++)
                        {
                                pcheck[i].input(cin);
                        }
                        //sums up cashed checks
                        for (i = 0; i < num_checks; i++)
                        {
                                if (pcheck[i].get_cashed())
                                {
                                        check_total = (pcheck[i].get_amount()).add(che
ck_total);
                                }
                        }
                        cout << divider << "Total from cashed checks:  ";
                        check_total.output(cout);
                        cout << endl;
                }
        }
        else // failed to allocate space
        {
                cout << "Unable to allocate space." << endl;
                cout << "Please shut down other applications first before running"
                        "program.\n";
                has_space = false;
        }
        if (has_space)
        {
                long num_deposits;
                cout << divider << "How many deposits will you enter? ";
                cin >> num_deposits;
                while (num_deposits < 0)
                {
                        cout << "Invalid number.\nHow many deposits will you enter? ";
                        cin >> num_deposits;
                }
                Money our_balance;
                Money deposit_total;
                if (num_deposits == 0) // user wants to make no deposits
                {
                        //Balance = Last balance - cashed checks
                        our_balance = last_balance.subtract(check_total);
                        cout << divider << "Calculated balance: ";
                        our_balance.output(cout);
                        cout << endl;
                        cout << divider << "Difference between calculated and reported
"
                                "balance: ";
                        our_balance.subtract(bank_balance).output(cout);
                        cout << endl;

                }

                else
                {
                        Money *pdeposit = NULL;
                        // allocates space for array of deposits
                        pdeposit = new Money[num_deposits];
                        if (pdeposit != NULL) //successfully allocated space
                        {
                                cout << divider << "Enter deposit amounts separated by
 spaces"
                                        "or ENTER key.\n";
                                //inputs deposits
                                for (i = 0; i < num_deposits; i++)
                                {
                                        pdeposit[i].input(cin);
                                }
                                //sums deposits
                                for (i = 0; i < num_deposits; i++)
                                {
                                        deposit_total = (pdeposit[i]).add(deposit_tota
l);
                                }
                                cout << divider << "Total from deposits:  ";
                                deposit_total.output(cout);
                                cout << endl;
                                //Balance = Last balance +deposits - cashed checks
                                our_balance = deposit_total.add
                                (last_balance.subtract(check_total));
                                cout << divider << "Calculated balance: ";
                                our_balance.output(cout);
                                cout << endl;
                                cout << divider << "Difference between calculated and"
                                        "reported balance: ";
                                our_balance.subtract(bank_balance).output(cout);
                                cout << endl;
                                //frees up used space
                                delete[] pdeposit;
                                pdeposit = NULL;
                        }
                        else // couldnt make space
                        {
                                cout << "Unable to allocate space." << endl;
                                cout << "Please shut down other applications first bef
ore"
                                        "running program.\n";
                                has_space = false;
                        }

                }
                if (has_space)
                {
                        //Bubble sort checks by increasing order of check#
                        for (long j = 0; j < num_checks; j++)
                        {
                                for (long k = 0; k < num_checks - 1; k++)
                                        if (pcheck[k].get_number() > pcheck[k + 1].get
_number())
                                        {
                                                Swap(pcheck[k], pcheck[k + 1]);
                                        }
                        }
                        //Outputs cashed checks in order
                        cout << divider << "Cashed checks from lowest to highest"
                                "check number:\n";
                        for (i = 0; i < num_checks; i++)
                        {
                                if (pcheck[i].get_cashed())
                                {
                                        cout << "Check #" << pcheck[i].get_number() <<
 ": ";
                                        pcheck[i].get_amount().output(cout);
                                        cout << endl;
                                }
                        }
                        //Outputs uncashed checks in order
                        cout << divider << "Uncashed checks from lowest to highest"
                                "check number:\n";
                        for (i = 0; i < num_checks; i++)
                        {
                                if (!(pcheck[i].get_cashed()))
                                {
                                        cout << "Check #" << pcheck[i].get_number() <<
 ": ";
                                        pcheck[i].get_amount().output(cout);
                                        cout << endl;
                                }
                        }
                }

        }
        //frees up used space
        delete[] pcheck;
        pcheck = NULL;

        cout << "\nPress q to quit." << endl;
        cin.ignore(INT_MAX, ’q’);
        return 0;

}
\033]0;p_stach@mars:˜/Balance\007[p_stach@mars Balance]$ cat Check.cpp
#include "Check.h"
#include "Money.h"
#include <iostream>

using namespace std;
//Function to input data into check
void Check::input(std::istream & ins)
{
        char choice;
        ins >> number;
        amount.input(ins);
        ins >> choice;
        //returns true or false depending on y/n
        if (choice == ’y’)
        {
                cashed = true;
        }
        else
        {
                cashed = false;
        }

}
\033]0;p_stach@mars:˜/Balance\007[p_stach@mars Balance]$ cat Check.h

#ifndef CHECK_H
#define CHECK_H
//Check Implementation File
#include <iostream>
#include "Money.h"

class Check
{
        //member variables
        long number;
        Money amount;
        bool cashed;

public:
        // Constructor
        Check() : number(), amount(), cashed() {}
        //Accessors
        long get_number() { return number; }
        Money get_amount() { return amount; }
        bool get_cashed() { return cashed; }
        //Mutator
        void input(std::istream & ins);





};

#endif
\033]0;p_stach@mars:˜/Balance\007[p_stach@mars Balance]$ cat Money.cpp
#include "Money.h"
#include <cmath>
#include <iomanip>
#include <iostream>
#include <climits>
using namespace std;

Money Money::add(const Money & amount) const
{
        long new_cents = cents + amount.get_cents();
        long new_dollars = dollars + amount.get_dollars();
        // checks if has excess cents in calculations
        if (new_cents > 99)
        {
                new_dollars++;
                new_cents = new_cents - 100;
        }
        return Money(new_dollars, new_cents);
}

Money Money::subtract(const Money & amount) const
{
        long new_cents = cents - amount.get_cents();
        long new_dollars = dollars - amount.get_dollars();
        // checks if does not have enough cents in calculations
        if (new_cents < 0)
        {
                new_dollars--;
                new_cents = new_cents + 100;
        }
        return Money(new_dollars, new_cents);
}


Money Money::negate() const
{
        return Money(-1 * dollars, cents);
}

bool Money::equals(const Money & amount) const
{
        bool rv = false;
        if (cents == amount.get_cents() && dollars == amount.get_dollars())
        {
                rv = true;
        }
        return rv;
```

```
}

bool Money::less(const Money & amount) const
{
        bool rv = false;
        if ((dollars+ .01*cents) < (amount.get_dollars() + .01* amount.get_cents()))
        {
                rv = true;
        }
        return rv;
}
void Money::input(std::istream & ins)
{
        char dummy;
        ins >> dummy >> dollars >> dummy >> cents;


}

void Money::output(std::ostream & outs) const
{
        outs << ’$’ << dollars << ’.’ << setfill(’0’) << setw(2) << cents;


}

double Money::get_value(void) const
{

        return  (dollars + cents*.01);
}
\033]0;p_stach@mars:˜/Balance\007[p_stach@mars Balance]$ cat Money.h
// This is the HEADER FILE money.h.  This is the INTERFACE for the class
// Money.  Values of this type are amounts of money in U.S. currency.

#ifndef MONEY_H
#define MONEY_H

#include <iostream>

class Money
{
        long cents;   // Cents
        long dollars;

public:

        // Initializes the object to $0.00.
        Money(void) : cents(0), dollars(0) {}

        // Initializes the object with a dollar value
        Money(long i_dollars) : cents(0), dollars(i_dollars) {}

        // Initializes the object with a dollar and cent value.
        Money(long i_dollars, short i_cents) : cents(i_cents), dollars(i_dollars) {}


        // Accessor for cents
        long get_cents() const { return cents; }
        // Accessor for dollars
        long get_dollars()  const { return dollars; }
        // Postcondition:  return value is sum of calling object and amount.
        //                 neither amount nor calling object are changed.
        Money add(const Money & amount) const;

        // Postcondition:  return value is difference of calling object and amount.
        //                 neither amount nor calling object are changed.
        Money subtract(const Money & amount) const;

        // Postcondition:  return value is arithmetic negation of calling object.
        //                 calling object is not changed.
        Money negate(void) const;

        // Returns true if the calling object equals the amount, false otherwise.
        bool equals(const Money & amount) const;

        // Returns true if the calling object is less than the amount,
        // false otherwise.
        bool less(const Money & amount) const;

        // Postcondition:  calling object’s value is read from the stream
        //                 in normal U.S. format:  $ddd.cc.
        void input(std::istream & ins);

        // Postcondition:  calling object’s value is printed on the stream
        //                 in normal U.S. format:  $ddd.cc.  (calling object
        //                 is not changed)
        void output(std::ostream & outs) const;

        // Returns amount of money in decimal format.
        double get_value(void) const;
};

#endif
\033]0;p_stach@mars:˜/Balance\007[p_stach@mars Balance]$ cat data.txt
9 $35.25 y
3 $0.25 y
4 $1.00 y
5 $2.75 n
7 $100.00 n
10 $20.00 n
\033]0;p_stach@mars:˜/Balance\007[p_stach@mars Balance]$ CPP Check Money main
Check.cpp...
Money.cpp...
main.cpp***


\033]0;p_stach@mars:˜/Balance\007[p_stach@mars Balance]$ ./main.out
Welcome to the Checkbook Balancing Program
Cash amounts are entered in the form [$ddd.cc]
------------------------------------------------------------
Enter last month’s balance: $500.00
------------------------------------------------------------
Enter the balance reported by bank: $370.05
------------------------------------------------------------
How many checks will you enter? 6
------------------------------------------------------------
Input from file? y/n: y
Enter file name: data.txt]
File does not exist.
Enter file name: data.txt
’data.txt’ selected as input file.
------------------------------------------------------------
Total from cashed checks:  $36.50
------------------------------------------------------------
How many deposits will you enter? 3
------------------------------------------------------------
Enter deposit amounts separated by spacesor ENTER key.
```

```
$20.00
$35.25
$39.85
$87.47
$10.98
$34.90
------------------------------------------------------------
Total from deposits:  $228.45
------------------------------------------------------------
Calculated balance: $691.95
------------------------------------------------------------
Difference between calculated andreported balance: $321.70
------------------------------------------------------------
Cashed checks from lowest to highestcheck number:
Check #3: $0.25
Check #4: $1.00
Check #9: $35.25
------------------------------------------------------------
Uncashed checks from lowest to highestcheck number:
Check #5: $2.75
Check #7: $100.00
Check #10: $20.00

Press q to quit.
q
\033]0;p_stach@mars:~/Balance\007[p_stach@mars Balance]$ ./main.out
Welcome to the Checkbook Balancing Program
Cash amounts are entered in the form [$ddd.cc]
------------------------------------------------------------
Enter last month's balance: $359.00
------------------------------------------------------------
Enter the balance reported by bank: $200.50
------------------------------------------------------------
How many checks will you enter? 5
------------------------------------------------------------
Input from file? y/n: n
Enter data in the format (without braces):
[Check Number] [Check Amount] [Cashed? y/n]
4 $42.00 y
7 $39.45 y
2 $29.61n
9 $49.17 y
1 $42.39 n
------------------------------------------------------------
Total from cashed checks:  $130.62
------------------------------------------------------------
How many deposits will you enter? 0
------------------------------------------------------------
Calculated balance: $228.38
------------------------------------------------------------
Difference between calculated and reportedbalance: $27.88
------------------------------------------------------------
Cashed checks from lowest to highestcheck number:
Check #4: $42.00
Check #7: $39.45
Check #9: $49.17
------------------------------------------------------------
Uncashed checks from lowest to highestcheck number:
Check #1: $42.39
Check #2: $29.61

Press q to quit.
q
\033]0;p_stach@mars:~/Balance\007[p_stach@mars Balance]$ exit
exit

Script done on Tue 31 Oct 2017 03:40:27 AM CDT
```