```
Script started on 2021-04-22 18:57:36-0500
m_sadaf1@ares:~$ pwd
/home/students/m_sadaf1
m_sadaf1@ares:~$ cat balance.cpp
#include "balance_money.h"
#include "balance_check.h"
#include <string>
#include <iostream>
#include <limits>
#include <fstream>

using namespace std;

// Function to swap checks

inline void Swap(Check &y, Check&z)
{
    Check temp = y;
    y = z;
    z = temp;
}

int main()
{
    cout << "\t\t\tWelcome to the Checkbook"
            "Balancing Program\n";
    cout << "Please enter the amount in [$ddd.cc]\n";

    cout << "Enter previous balance: ";
    bool space = true; // Checks if enough space for array
    Money pre_balance;
    pre_balance.input(cin);
    cout << "Enter the balance given by bank: ";
    Money bank_balance;
    bank_balance.input(cin);
    Check *pcheck = nullptr; // Pointer for array
    long i, num_checks;
    cout << "How many checks will you enter? ";
    cin >> num_checks;
    while (num_checks < 1)
    {
        cout << "Error. \nHow many checks will you enter? ";
        cin >> num_checks;
    }
    pcheck = new Check[num_checks];
    Money check_total;
    if (pcheck != NULL)
    {
        char choice;
        cout << "Import from file? y/n: ";
        cin >> choice;
        while (choice != 'y' && choice != 'n')
        {
            cout << "Error.\nImport from file? y/n: ";
            cin >> choice;
        }

        // Import from file

        if (choice == 'y')
        {
            ifstream input;
            string filename;

            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "enter file name: ";

            // Checks if the file exists

            getline(cin, filename);
            input.open(filename);
            while (!input)
            {
                input.close();
                input.clear();

                cout << "File not found!\nEnter file name: ";
                getline(cin, filename);
                input.open(filename);
            }
            cout << filename << "selected.";

            // Inputs checks into array

            for (i = 0; i < num_checks; i++)
            {
                pcheck[i].input(input);
            }

            // Totals cashed checks

            for (i = 0; i < num_checks; i++)
            {
                if (pcheck[i].get_cashed())
```

```
            {
                check_total = (pcheck[i].get_amount())
                        .add(check_total);
            }
        }
        cout << "Total cashed checks: ";
        check_total.output(cout);
        input.close();
    }
    else
    {
        // Input from Keyboard

        cout << "Enter in the format WITHOUT BRACES:\n"
                "[Check Number] [Check Amount]"
                "[Cashed? y/n]\n";

        // Inputs checks into array

        for (i = 0; i < num_checks; i++)
        {
            pcheck[i].input(cin);
        }

        // Totals cashed checks

        for (i = 0; i < num_checks; i++)
        {
            if (pcheck[i].get_cashed())
            {
                check_total = (pcheck[i].get.amount())
                        .add(check_total);
            }
        }

        cout << "Total cashed checks: ";
        check_total.output(cout);
    }
}
else
{
    cout << "Unable to allocate space.";
    cout << "Please close other applications first.\n";
    space = false;
}
if (space)
{
    long num_deposits;
    cout << "How many deposits will you enter? ";
    cin >> num_deposits;
    while (num_deposits < 0)
    {
        cout << "Error.\nHow many deposits will you enter? ";
        cin >> num_deposits;
    }
    Money balance;
    Money deposit_total;
    if (num_deposits == 0) // No deposit by the user
    {
        balance = pre_balance.subtract(check_total);
        cout << "Calculated balance: ";
        balance.output(cout);
        cout << "Difference between calculated and "
                "bank balance: ";
        balance.subtract(bank_balance).output(cout);
    }
    else
    {
        Money *pdeposit = nullptr;

        // Allocates space for array

        pdeposit = new Money[num_deposits];
        if (pdeposit != nullptr) // Space allot successful
        {
            cout << "Enter deposit amounts with spaces "
                    "in between or ENTER key.\n";

            // Inputs deposists

            for (i = 0; i < num_deposits; i++)
            {
                pdeposit[i].input(cin);
            }

            // total deposits

            for (i = 0; i < num_deposits; i++)
            {
                deposit_total = (pdeposit[i])
                        .add(deposit_total);
            }
            cout << "Total deposits: ";
            deposit_total.output(cout);
            balance = deposit_total.add
                    (pre_balance.subtract(check_total));
```

```
            cout << "Calculated balance: ";
            balance.output(cout);
            cout << "difference between calculated and "
                    "bank balance: ";
            balance.subtract(bank_balance).output(cout);

            // To free up used space

            delete[] pdeposit;
            pdeposit = nullptr;
        }
        else
        {
            cout << "Unable to allocate space. ";
            cout << "Please close other applications first.\n";
            space = false;
        }
    }

    if (space)
    {
        // Bubble sort by increasing order

        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]);
            }
        }

        // Output cash in order

        cout << "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);
            }
        }

        // Outputs uncashed checks in order

                cout << "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].getnumber() << ": ";
                        pcheck[i].get_amount().output(cout);
                    }
                }
            }
        }

        delete[] pcheck;
        pcheck = nullptr;

        cout << "\n Thank you for using CBP!! "
                "Have a great day!";
        cout << "\nPress q to quit the program.";
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return 0;
    }

    m_sadaf1@ares:~$ cat balance.txt
    9 $3.55 y
    4 $17.99 n
    5 $100.00 n
    1 $8.88 y
    3 $16.00 n
    7 $98.99 n
    2 $150.90 y
    6 $50.00 y
    8 $33.33 n

    m_sadaf1@ares:~$ cat balance_check.cpp
    #include "balance_check.h"
    #include "balance_money.h"
    #include <iostream>

    using namespace std;

    void Check::input(std::istream & ins) // Input data in check
    {
        char choice;
        ins >> number;
        amount.input(ins);
```

```
    ins >> choice;

    if (choice == 'y') // returns true or false
    {
        cashed = true;
    }
    else
    {
        cashed = false;
    }
}

m_sadaf1@ares:~$ cat balance_check.h
#ifndef BALANCE_CHECK_H
#define BALANCE_CHECK_H

#include <iostream>
#include "balance_money.h"

class Check
{
    // Member Variables
    long number;
    Money amount;
    bool cashed;

public:   // Constructors
        Check() : number(), amount(), cashed() {}

        // Accessors

        long get_number() const { return number; }
        Money get_amount() const { return amount; }
        bool get_cashed() const { return cashed; }

        // Mutator

        void input(std::istream & ins);
};

#endif /*BALANCE_CHECK_H*/

m_sadaf1@ares:~$ cat balance_cheoney
#include "balance_money.h"
#include <cmath>
#include <iomanip>
#include <iostream>
#include <limits>


using namespace std;

Money Money::add(const Money & amount) const
{
    short new_cents = cents + amount.get_cents();
    long new_dollars = dollars +amount.get_dollars();

    // To check if 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
{
    short new_cents = cents - amount.get_cents();
    long new dollars = dollars - amount.get_dollars();

    // If not enough cents in calculation

    if (new_cents < 0)
    {
        new_dollars--;
        new_cents = new_cents + 100;
    }
    return Money(new_dollars, new_cents);
}

Money Money::negate() const
{
    return Money(-get_dollars, get_cents);
}

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

```
bool Money::less(const Money & amount) const
{
    bool s = false;
    if ((dollars + .01*cents) < (amount.get_dollars()
            + .01* amount.get_cents()))
    {
        s = true;
    }
    return s;
}

void Money::input(cin & ins)
{
    char u;
    ins >> u >> dollars >> u >> cents;
}

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

double Money::get_value(void) const
{
    return static_cast<double>(all_cents / 100);
}


m_sadaf1@ares:~$ cat balance_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 all_cents;  // monetary value stored as pennies

public:
        // Initializes the object to $0.00.
        Money(void) : all_cents(0) {};

        // Initializes the object to dollars*100 cents.
        Money(long i_dollars) :
                all_cents(i_dollars * 100) {};

        // Initializes the object to dollars*100 + cents.
        Money(long i_dollars, short i_cents) :
                all_cents(i_dollars * 100 + i_cents) {};

        // Accessor for cents
        short get_cents() const { return static_cast<short>
                (all_cents % 100); }

        // Accessor for dollars
        long get_dollars() const { return
                (all_cents / 100); }

        // 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

m_sadaf1@ares:~$ CPP balance_check balance_money balance
balance.cpp***
balance_check.cpp...
balance_money.cpp...
balance.cpp: In function ‘int main()’:
balance.cpp:122:46: error:
‘class Check’ has no member named
‘get’
                    check_total =
                    (pcheck[i].get.amount())
                                             ^~~
balance.cpp:241:45: error: expected
primary-expression before ‘)’ token
                if (!pcheck[i].get_cashed()))
                                             ^
balance_money.cpp: In member function ‘Money
Money::add(const Money&) const’:
balance_money.cpp:11:23: error:
‘cents’ was not declared in this scope
     short new_cents = cents + amount.get_cents();
                       ^~~~
balance_money.cpp:12:24: error:
‘dollars’ was not declared in this scope
     long new_dollars = dollars +amount.get_dollars();
                        ^~~~~~
balance_money.cpp:12:24: note: suggested
alternative: ‘llabs’
     long new_dollars = dollars +amount.get_dollars();
                        ^~~~~~
                        llabs
balance_money.cpp:19:31: warning:
conversion to ‘short int’
from ‘int’ may alter its value
[-Wconversion]
         new_cents = new_cents - 100;
                     ~~~~~~~~~~^~~~
balance_money.cpp: In member function ‘Money
Money::subtract(const Money&) const’:
balance_money.cpp:26:23: error:
‘cents’ was not declared in this scope
     short new_cents = cents - amount.get_cents();
                       ^~~~
balance_money.cpp:27:10: error: expected
unqualified-id before ‘new’
     long new dollars = dollars -
     amount.get_dollars();
          ^~~
balance_money.cpp:33:9: error:
‘new_dollars’ was not declared in this scope
         new_dollars--;
         ^~~~~~~~~~
balance_money.cpp:33:9: note: suggested
alternative: ‘get_dollars’
         new_dollars--;
         ^~~~~~~~~~
         get_dollars
balance_money.cpp:34:31: warning:
conversion to ‘short int’
from ‘int’ may alter its value
[-Wconversion]
         new_cents = new_cents + 100;
                     ~~~~~~~~~~^~~~
balance_money.cpp:36:18: error:
‘new_dollars’ was not declared in this scope
     return Money(new_dollars, new_cents);
                  ^~~~~~~~~~
balance_money.cpp:36:18: note: suggested
alternative: ‘get_dollars’
     return Money(new_dollars, new_cents);
                  ^~~~~~~~~~
                  get_dollars
balance_money.cpp: In member function ‘Money
Money::negate() const’:
balance_money.cpp:41:19: error: wrong
type argument to unary minus
     return Money(-get_dollars, get_cents);
                   ^~~~~~~~~~
balance_money.cpp: In member function ‘bool
Money::equals(const Money&) const’:
balance_money.cpp:47:9: error:
‘cents’ was not declared in this scope
     if (cents == amount.get_cents() && dollars ==
         ^~~~
balance_money.cpp:47:40: error:
‘dollars’ was not declared in this scope
     if (cents == amount.get_cents() && dollars ==
                                        ^~~~~~
balance_money.cpp:47:40: note: suggested
alternative: ‘llabs’
     if (cents == amount.get_cents() && dollars ==
                                        ^~~~~~
```

```
                                        llabs
balance_money.cpp: In member function ‘bool
Money::less(const Money&) const’:
balance_money.cpp:58:10: error:
‘dollars’ was not declared in this scope
     if ((dollars + .01*cents) < (amount.get_dollars()
          ^~~~~~~
balance_money.cpp:58:10: note: suggested
alternative: ‘llabs’
     if ((dollars + .01*cents) < (amount.get_dollars()
          ^~~~~~~
          llabs
balance_money.cpp:58:24: error:
‘cents’ was not declared in this scope
     if ((dollars + .01*cents) < (amount.get_dollars()
                        ^~~~~
balance_money.cpp:58:52: warning:
conversion to ‘double’ from ‘long
int’ may alter its value [-Wconversion]
     if ((dollars + .01*cents) < (amount.get_dollars()
                                  ~~~~~~~~~~~~~~~~~^~
balance_money.cpp: At global scope:
balance_money.cpp:66:19: error: variable
or field ‘input’ declared void
 void Money::input(cin & ins)
                   ^~~
balance_money.cpp:66:25: error:
‘ins’ was not declared in this scope
 void Money::input(cin & ins)
                         ^~~
balance_money.cpp:66:25: note: suggested
alternative: ‘int’
 void Money::input(cin & ins)
                         ^~~
                         int
balance_money.cpp: In member function ‘void
Money::output(std::ostream&) const’:
balance_money.cpp:74:20: error:
‘dollars’ was not declared in this scope
     outs << '$' << dollars << setfill('0') << setw(2)
                    ^~~~~~~
balance_money.cpp:74:20: note: suggested
alternative: ‘llabs’
     outs << '$' << dollars << setfill('0') << setw(2)
                    ^~~~~~~
                    llabs
balance_money.cpp:75:16: error:
‘cents’ was not declared in this scope
```

```
               << cents;
                  ^~~~~


m_sadaf1@ares:~$ exit
exit

Script done on 2021-04-22 18:59:43-0500
```