```
Script started on Sun 09 Apr 2017 11:21:00 PM CDT
\033]0;b_pepa@mars:~/CSC122/balance\007[b_pepa@mars balance]$ cat balance.info
Brandon Pepa
CSC122-001
Balance in all things...
Project

(level 4.5)
(level 2)
       Overload operators for check class
(level 1)
       overload operators for Money class

**(level 7.5)**

Description:
       The function of this program is to be able to record checks, add
deposits, and find the new balance of an account.
\033]0;b_pepa@mars:~/CSC122/balance\007[b_pepa@mars balance]$ cat p\033[K033[K033[Kbalance.
cpp\033[K033[K033[K033[K033[K033[K033[K033[K033[K033[K033[K033[K\007
balance.cpp   balance.out   money.h
balance.info  money.cpp     typescript
[b_pepa@mars balance]$ cat
balance.cpp   balance.out   money.h
balance.info  money.cpp     typescript
[b_pepa@mars balance]$ cat money.h
#ifndef MONEY_H
#define MONEY_H

#include <iostream>

class Money
{
   long all_cents;
public:

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

   //Initializes the object to dollars*100 cents.
   Money(long dollars, short cents = 0) :
      all_cents(dollars*100 + cents) {}


   Money operator+(const Money & m) const
      { Money t(*this);
        t.all_cents += m.all_cents;
        return t; }

   Money operator-(const Money & m) const
      { Money t(*this);
        t.all_cents -= m.all_cents;
        return t; }

   Money operator-(void) const
      { Money t(*this);
        t.all_cents = -t.all_cents;
        return t; }

   Money & operator+=(const Money & m)
      { this->all_cents += m.all_cents;
        return *this; }

   Money & operator-=(const Money & m)
      { this->all_cents -= m.all_cents;
        return *this; }

   bool operator==(const Money & m) const
      { return (all_cents == m.all_cents); }
   bool operator!=(const Money & m) const
      { return (all_cents != m.all_cents); }
   bool operator<=(const Money & m) const
      { return (all_cents <= m.all_cents); }
   bool operator>=(const Money & m) const
      { return (all_cents >= m.all_cents); }
   bool operator<(const Money & m) const
      { return (all_cents < m.all_cents); }
   bool operator>(const Money & m) const
      { return (all_cents < m.all_cents); }


   void input(std::istream & ins = std::cin);
   void output(std::ostream & outs = std::cout) const;

   friend std::istream & operator>>(std::istream & ins, Money & m)
      { m.input(ins); return ins; }

   friend std::ostream & operator<<(std::ostream & outs, const Money & m)
      { m.output(outs); return outs; }

   double get_value(void) const
      { return all_cents/100.0; }
};



#endif
\033]0;b_pepa@mars:~/CSC122/balance\007[b_pepa@mars balance]$ cat money.cpp
#include "money.h"

#include <iostream>

void Money::input(std::istream & ins)
{
   char dummy;
   long dollars, cents;
   ins >> dummy >> dollars >> dummy >> cents;
   all_cents = dollars*100 + cents;
return;
}

void Money::output(std::ostream & outs) const
{
   outs << '$' << all_cents/100 << '.'
        << (all_cents%100 < 10 ? '0' : '\0')
        << all_cents%100;
return;
}
\033]0;b_pepa@mars:~/CSC122/balance\007[b_pepa@mars balance]$ cat balance.cpp
#include "money.h"
#include <iostream>
#include <fstream>
#include <cctype>
using namespace std;


class check
{
```

```
   long number;
   Money amount;
   bool cashed;
public:

   check(void) : number(0), amount(0), cashed(false) {}
   check(const check & o) : number(o.number), amount(o.amount),
                     cashed(o.cashed) {}
   check(long n, Money a, bool c) : number(n), amount(a),
                                    cashed(c) {}

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

   bool set_number(const long & n);
   bool set_amount(const Money & a);
   bool set_cashed(const bool & c);

   void output(std::ostream & os = std::cout) const;
   void input(std::istream & is = std::cin);

   bool operator>(const check & c) const;

   friend Money operator+(const Money & m, const check & c)
      { return m + c.get_amount(); }
   friend Money operator-(const Money & m, const check & c)
      { return m - c.get_amount(); }

   friend std::ostream & operator<<(std::ostream & out, const check & c)
      { c.output(out); return out; }
   friend std::istream & operator>>(std::istream & in, check & c)
      { c.input(in); return in; }


};

void sort(check * & p, size_t size);
inline void swap(check & x, check & y)
{
   check t(x);
   x = y;
   y = t;
   return;
}

inline Money find_nBal(Money orig, Money check_tot, Money dep_tot)
   { return orig + dep_tot - check_tot; }


int main(void)
{
   //Variables associated with the dynamic array for checks
   check * pbook;
   size_t booksize, bookcur = 0;
   Money ctot;

   //Variables associated with the dynamic array for deposits
   Money * pdep;
   size_t depsize, depcur = 0;
   Money dtot;

   //last balance(lBal) and new balance(nBal)
   Money lBal, nBal;



   cout << "\tWelcome to the Check Balancing Program!";

   cout << "\n\nHow many checks do you have to enter? ";
   cin >> booksize;

   pbook = new check [booksize];
   if(pbook == NULL)
   {
      cerr << "\n\aUnable to allocate space for "
           << booksize << " values!\n\n"
           << "Please shut down other applications first...";

      //might as well exit the program if we weren't able to allocate
      //this memory becuase like... the user needs to fix it...
      return 0;
   }

   for(size_t i = bookcur; i < booksize; ++i)
   {
      cin >> *(pbook + i);
   }

   cout << "\n\nPlease enter the Last balance on this account\n";
   cin >> lBal;

   cout << "\n\nHow many deposits do you have to enter? ";
   cin >> depsize;

   pdep = new Money [depsize];
   if(pdep == NULL)
   {
      cerr << "\n\aUnable to allocate space for "
           << depsize << " values!\n\n"
           << "Please shut down other applications first...";

      //might as well exit the program if we weren't able to allocate
      //this memory becuase like... the user needs to fix it...
      return 0;
   }

   for(size_t i = 0; i < depsize; ++i)
   {
      cout << "Enter amount for deposit " << i + 1 << "."
           << ((i == 0) ? "(in $00.00 format) " : " ");
      cin >> *(pdep + i);
   }

   sort(pbook, booksize);

   for(size_t i = depcur = 0; i < depsize; ++i)
   {
      dtot += *(pdep + i);
   }

   bookcur = 0;
   while(bookcur < booksize && pbook[bookcur].get_cashed())
   {
      ctot += pbook[bookcur].get_amount();
```

```
      ++bookcur;
   }

   nBal = find_nBal(lBal, ctot, dtot);

   while(bookcur < booksize)
   {
      ctot += pbook[bookcur].get_amount();
      ++bookcur;
   }

   bookcur = 0;


   cout << "\n\n" << string(60,'*')
        << "\nThe total for all checks is: " << ctot
        << "\nThe total for all deposits is: " << dtot
        << "\nYour new balance is: " << nBal;

   cout << "\n\n***Cashed Checks***"
        << "\nCheck #\tamount";

   while(bookcur < booksize && pbook[bookcur].get_cashed())
   {
      cout << '\n' << *(pbook + bookcur);
      ++bookcur;
   }

   cout << "\n\n***Uncashed Checks***"
        << "\nCheck #\tamount";

   while(bookcur < booksize)
   {
      cout << '\n' << *(pbook + bookcur);
      ++bookcur;
   }


   delete [] pbook;
   delete [] pdep;
   pbook = NULL;
   pdep = NULL;
return 0;
}

bool check::operator>(const check & c) const
{
   check t(*this);
   //non cashed checks will ALWAYS be greater than cashed checks
   if(t.cashed != c.cashed)
   {
      return !t.cashed;
   }
   return t.number > c.number;
}

bool check::set_number(const long & n)
{
   if(n >= 0)
   {
      number = n;
      return true;
   }
   return false;
}

bool check::set_amount(const Money & a)
{
   amount = a;
   return true;
}

bool check::set_cashed(const bool & c)
{
   cashed = c;
   return true;
}

void check::output(ostream & os) const
{
   check c(*this);
   os << c.number << '\t' << c.amount;
   return;
}

void check::input(istream & is)
{
   char YorN;
   is >> number >> amount >> YorN;
   cashed = (toupper(YorN) == 'Y') ? true : false;
}

void sort(check * & p, size_t size)
{
   bool done = false;
   size_t i = 0;
   while(i < size && !done)
   {
      done = true;
      for(size_t j = 0; j + i + 1 < size; ++j)
      {
         if(p[j] > p[j+1])
         {
            swap(p[j], p[j+1]);
            done = false;
         }
      }
      ++i;
   }
   return;
}
\033]0;b_pepa@mars:~/CSC122/balance\007[b_pepa@mars balance]$ CPP balance money
balance.cpp***
money.cpp...


\033]0;b_pepa@mars:~/CSC122/balance\007[b_pepa@mars balance]$ X033[K./blan033[K033[K
\033[Kalacne033[K033[K033[Kncla033[K033[Ke.out
        Welcome to the Check Balancing Program!

How many checks do you have to enter? 10
1001 $15.00 y
1003 $20.00 n
1002 $40.00 n
1007 $132.15 n
1004 $168.89 y
1006 $95.87 y
1005 $63.86 n
```

```
1001 $43.12 n
1002 $12.98 y
1020 $76.00 n


Please enter the Last balance on this account
$800.45


How many deposits do you have to enter? 6
Enter amount for deposit 1.(in $00.00 format) $50.00
Enter amount for deposit 2. $90.00
Enter amount for deposit 3. $75.00
Enter amount for deposit 4. $100.00
Enter amount for deposit 5. $45.00
Enter amount for deposit 6. $75.00


************************************************************
The total for all checks is: $664.\00057
The total for all deposits is: $435.00
Your new balance is: $835.\00071

***Cashed Checks***
Check # amount
1001    $15.00
1004    $165.\00089
1006    $95.\00087
1032    $12.\00098

***Uncashed Checks***
Check # amount
1002    $40.00
1003    $20.00
1005    $63.\00056
1007    $132.\00015
1011    $43.\00012
1020    $76.00\033]0;b_pepa@mars:~/CSC122/balance\007[b_pepa@mars balance]$ exit
exit

Script done on Sun 09 Apr 2017 11:25:30 PM CDT
```