```
#ifndef _VIEW_HELPER_HPP_
#define _VIEW_HELPER_HPP_

#include "transaction.hpp"
#include "account.hpp"
#include "centerhelper.hpp"

void printBanner() {
  std::cout << std::setfill('-');
  std::cout << std::setw(80) << centered(R"x(--------------------------------)x") << "
\n";
  std::cout << std::setw(80) << centered(R"x( __  __                          )x") << "
\n";
  std::cout << std::setw(80) << centered(R"x(|  \/  |                         )x") << "
\n";
  std::cout << std::setw(80) << centered(R"x(| \  / | ___  _ __   ___ _   _  )x") << "
\n";
  std::cout << std::setw(80) << centered(R"x(| |\/| |/ _ \| '_ \ / _ \ | | | )x") << "
\n";
  std::cout << std::setw(80) << centered(R"x(| |  | | (_) | | | |  __/ |_| | )x") << "
\n";
  std::cout << std::setw(80) << centered(R"x(|_|  |_|\___/|_| |_|\___|\__, | )x") << "
\n";
  std::cout << std::setw(80) << centered(R"x(                          __/ | )x") << "
\n";
  std::cout << std::setw(80) << centered(R"x(                         |___/  )x") << "
\n";
  std::cout << std::setw(80) << centered(R"x(--------------------------------)x") << "
\n";
  std::cout << std::setfill(' ');
  std::cout << "\n\n\n\n\n\n\n\n\n\n\n";
}

std::string prompt() {
  std::string c;
  std::cout << "command> ";
  std::getline(std::cin, c);
  return c;
}

Transaction readTransaction(bool withdraw = false) {
  std::string account;
  long double amount;
  std::string memo;

  std::cout << std::setw(15) << std::left << "Account name" << std::right << ": ";
  std::cin >> account;
  std::cout << std::setw(15) << std::left << "Amount" << std::right << ": ";
  std::cin >> std::get_money(amount);
  std::cout << std::setw(15) << std::left << "Memo" << std::right << ": ";

  if (withdraw && amount > 0) {
    amount = -amount;
  }

  //This is needed to avoid issue after reading in a line after a
  //normal std::cin operation.
  std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
  std::getline(std::cin, memo);
  trim(memo);

  Transaction t(account, amount, memo);
  return t;
}

Account readAccount() {
  std::string accountName;

  std::cout << std::setw(20) << std::left << "New account name" << std::right << ": ";
```

```
  std::cin >> accountName;

  Account a(accountName);
  return a;
}

void deleteAccount(std::vector<Account>& accounts) {
  std::string accountName;
  std::cout << "Accounts avilable: \n";
  for (const Account& a : accounts) {
    std::cout << std::setw(30) << a.getName() << ": " << a.getTransactions().size()
      << " transactions\n";
  }

  std::cout << std::setw(15) << std::left << "Account name" << std::right << ": ";
  std::cin >> accountName;

  //Delete the account from the vector
  for (auto it = std::begin(accounts); it != std::end(accounts); ) {
    if (std::equal(std::begin((*it).getName()), std::end((*it).getName()), std::begin(
accountName))) {
      it = accounts.erase(it);
    }
    else {
      ++it;
    }
  }
}

void printHelp() {
  std::string line(80, '-');
  std::cout << std::setw(80) << centered("Commands") << "\n";
  std::cout << line << "\n";
  std::cout << std::setw(20) << "help" << std::setw(50) << "Print Help Message\n";
  std::cout << std::setw(20) << "new" << std::setw(50) << "Create a new account\n";
  std::cout << std::setw(20) << "deposit" << std::setw(50) << "Create a new deposit\n"
;
  std::cout << std::setw(20) << "widthdraw" << std::setw(50) << "Create a new withdraw
al\n";
  std::cout << std::setw(20) << "ledger" << std::setw(50) << "Print out account ledger
\n";
  std::cout << std::setw(20) << "delete" << std::setw(50) << "Delete an account and al
l transactions\n";
  std::cout << std::setw(20) << "quit" << std::setw(50) << "Quit the program and write
 the data file\n";
  std::cout << "\n\n\n";
}

void addToAccount(Transaction& t, std::vector<Account>& accounts) {
  std::for_each(std::begin(accounts), std::end(accounts), [&](Account& a) {
      std::string an = t.getAccountName();
      if (std::equal(std::begin(an), std::end(an), std::begin(a.getName()))) {
        a.addTransaction(t);
      }
    });
}

void printAccountLedger(const Account& a) {
  std::string line(80, '-');
  std::ostringstream oss;
  oss << "Ledger for " << a.getName();
  std::cout << "\n\n";
  std::cout << std::setw(80) << centered(oss.str()) << "\n";
  std::cout << line << "\n";
  for (const Transaction& t : a.getTransactions()) {
    std::time_t time = t.getTransactionDate();
    struct tm* date = std::localtime(&time);
    std::cout << std::setw(20) << std::left << std::put_time(date, "%D %r");
    std::cout << std::setw(40) << std::right << t.getMemo();
```

```
    std::cout << std::setw(20) << std::right << std::put_money(t.getAmount(), true);
    std::cout << "\n";
  }

  oss.str("");
  oss.clear();
  oss.imbue(std::locale(""));
  oss << "Total: $" << std::put_money(a.balance());
  std::string dashes(oss.str().size(), '-');
  std::cout << std::setw(80) << dashes << "\n";
  std::cout << std::setw(80) << oss.str() << "\n";
}

#endif
```