```
/* Filename: main.cpp
 * ------------------
 * Class:   CSCI 162 -- Assignment 2
 * Date:    Feb 13th, 2018
 * Author:  Darwin Jacob Groskleg
 *
 * Purpose: to produce an updated set of account balances, a statement of the
 * months transactions for each account and report on accounts with very high
 * or low balances at the end of the month. This is all done from a set of
 * account balances and a set of transactions for the month.
 *
 * Sample Ouput:
 * ------------------------------------------------------------------------------
 * ABNORMAL ACCOUNTS REPORT (below $0 or above $10k)
 * CNum Name                  Amount
 * ---- -------------  -------------
 * 6239 R. Pat Turner  $    10230.12
 * 6543 Patrick Riley  $    11219.78
 * 7654 Tom T. Peters  $      -20.33
 * ------------------------------------------------------------------------------
 *
 * Sample balanceout.txt
 * ------------------------------------------------------------------------------
 * 1123 John T. Smith  $       98.73
 * 1456 Daniel Smith   $      245.83
 * 2416 Shannon Ross   $     1250.12
 * 2468 Betty White    $     1928.24
 * 2832 Timothy Adams  $     2606.27
 * 3893 Barney Kent    $      630.82
 * 4444 Steve Burns    $     5837.29
 * 5426 Melissa Black  $     3869.92
 * 5555 Fred G. Jones  $        0.00
 * 6239 R. Pat Turner  $    10230.12
 * 6543 Patrick Riley  $    11219.78
 * 6847 Patrick Blair  $      384.82
 * 6983 Henry Presley  $      340.99
 * 7654 Tom T. Peters  $      -20.33
 * 7736 Sara Wilson    $     9993.28
 * 8762 Julie Anderson $     1684.23
 * 8764 Jessica Stone  $        1.00
 * 8975 Lisa Gregory   $     1938.83
 * 9324 Andrew York    $        0.98
 * 9876 Jane Grant     $     1837.45
 * ------------------------------------------------------------------------------
 *
 * Sample statementout.txt
 * ------------------------------------------------------------------------------
 *            First Antigonish Bank
 *
 * Statement for  John T. Smith
 * Customer Number 1123
 *
 * Item               Deposit   Withdrawl    Balance
 * -----------        -------   ---------   --------
 * Starting Balance:                          28.28
 *
 *  Cheque                          30.00      -1.72
 *  Overdraft                        5.00      -6.72
 *  Deposit              2.00                 -4.72
 *  Overdraft                        5.00      -9.72
 *  Payroll           1213.45                1203.73
 *  Withdrawl                      500.00     703.73
 *  Cheque                         800.00     -96.27
 *  Overdraft                        5.00    -101.27
 *  No Bk Dep          200.00                 98.73
 *
 * Final Balance:                             98.73
 *
 *
```

```
 70   *            First Antigonish Bank
 71   *
 72   * Statement for  Daniel Smith
 73   * Customer Number 1456
 74   *
 75   * Item               Deposit   Withdrawl    Balance
 76   * -----------        -------   ---------   --------
 77   * Starting Balance:                          245.83
 78   *
 79   *
 80   * Final Balance:                             245.83
 81   *
 82   *
 83   *            First Antigonish Bank
 84   *
 85   * Statement for  Shannon Ross
 86   * Customer Number 2416
 87   *
 88   * Item               Deposit   Withdrawl    Balance
 89   * -----------        -------   ---------   --------
 90   * Starting Balance:                         1248.34
 91   *
 92   *  Deposit            389.89                1638.23
 93   *  Withdrawl                       388.11   1250.12
 94   *
 95   * Final Balance:                            1250.12
 96   * ------------------------------------------------------------------------
 97   */
 98  #include <iostream>
 99  #include <fstream>
100  #include <string>
101  #include <iomanip>
102
103  using namespace std;
104
105  struct balanceT {
106      int customer_number;
107      string customer_name;
108      int amount;     // in cents
109  };
110
111  struct transactionT {
112      int customer_number;
113      string memo;
114      int amount;     // in cents
115  };
116
117  bool ReadBalanceRecord(ifstream &, balanceT &);
118  bool ReadTransactionRecord(ifstream &, transactionT &);
119  void WriteInDollars(ostream &, int , bool endline=true, int dollarw=8);
120  void DisplayBalanceRecord(balanceT);
121  void WriteBalanceRecord(ofstream &, balanceT);
122  bool WriteStatement(ofstream &, balanceT &);
123  void WriteStatementTransaction(ofstream &, transactionT);
124
125  int main() {
126      balanceT b;
127      ifstream balances("data/balance.txt");
128      ofstream statementout("statementout.txt");
129      ofstream balanceout("balanceout.txt");
130
131      if (balances.fail()) {
132          cout << "ERROR: can't open file balance.txt" << endl;
133          return -1;
134      }
135      if (statementout.fail()) {
136          cout << "ERROR: can't open file statementout.txt" << endl;
137          return -1;
138      }
```

```
139      if (balanceout.fail()) {
140          cout << "ERROR: can't open file balanceout.txt" << endl;
141          return -1;
142      }
143
144      // Report Header for task #5
145      cout << "ABNORMAL ACCOUNTS REPORT (below $0 or above $10k)\n"
146           << "CNum Name                 Amount\n"
147           << "---- -------------  -------------"
148           << endl;
149      while (ReadBalanceRecord(balances, b)) {
150          // Task #7
151          if (!WriteStatement(statementout, b)) return -1;
152          // Task #5 less than 0 or more than $1,000.00
153          if (b.amount < 0 || b.amount > 1000000) DisplayBalanceRecord(b);
154          // Task #6
155          WriteBalanceRecord(balanceout, b);
156      }
157
158      balances.close();
159      statementout.close();
160      balanceout.close();
161
162
163      return 0;
164  }
165
166  /* Function: WriteStatement
167   * Usage: if (WriteStatement(stmt, bal)) return -1;
168   * Side Effects: value of balance struct argument points to is updated.
169   * ------------------------------------------------------------------
170   * Writes the statement for a single customer, given their balance, to a file.
171   * If the transactions file can't be accessed or fails the function returns
172   * false.
173   * Fulfills task #7 on the assignment.
174   */
175  bool WriteStatement(ofstream &outfile, balanceT &b) {
176      transactionT t;
177      ifstream transactions("data/trans.txt");
178
179      if (transactions.fail()) {
180          cout << "ERROR: can't open file trans.txt" << endl;
181          return false;
182      }
183
184      // Statement Header
185      outfile << setw(34) << "First Antigonish Bank\n\n"
186          << "Statement for  " << b.customer_name << "\n"
187          << "Customer Number " << b.customer_number << "\n\n"
188          << "Item               Deposit   Withdrawl    Balance\n"
189          << "-----------        -------   ---------   --------\n"
190          << left << setw(38) << "Starting Balance: " << right;
191      WriteInDollars(outfile, b.amount);
192      outfile << endl;
193      while (ReadTransactionRecord(transactions, t)) {
194          if (t.customer_number == b.customer_number) {
195              WriteStatementTransaction(outfile, t);
196              b.amount += t.amount;
197              // print balance in right column
198              WriteInDollars(outfile, b.amount);
199              // Task #4
200              if (b.amount < 0) {
201                  transactionT od = { b.customer_number, "Overdraft", -500 };
202                  WriteStatementTransaction(outfile, od);
203                  b.amount += od.amount;
204                  // print balance in right col
205                  WriteInDollars(outfile, b.amount);
206              }
207          }
```

```
208      }
209      transactions.close();
210      outfile << left << setw(39) << "\nFinal Balance: " << right;
211      WriteInDollars(outfile, b.amount);
212      outfile << endl << endl;
213
214      return true;
215  }
216
217  /* Function: WriteInDollars
218   * Usage: WriteInDollars(cout, 10000);
219   * Defaults: endline=true, dollarw=8
220   * --------------------------------
221   * Given a number of cents, writes the dollar representation to some ostream
222   * (i.e. cout or ofstream). By default it always follows up with an endline
223   * and uses 8 characters to represent dollar digits (11 total), which is neede
224   * for numbers up to 99,999,999.99 .
225   */
226  void WriteInDollars(ostream &outstr, int cents, bool endline, int dollarw) {
227      outstr << right << setw(dollarw) << (cents / 100) << '.'
228             << setfill('0') << setw(2) << abs(cents % 100)
229             << setfill(' ');
230      if (endline) outstr << endl;
231  }
232
233  /* Function: ReadTransactionRecord
234   * Usage: while(ReadTransactionRecord(trans, t)
235   *           { do something with t; }
236   * Side Effects: value of transaction struct pointed to in arguments is change
237   * ------------------------------------------------------------------------
238   * This function read transaction records from a file and assigns the values t
239   * the proper attributes of a struct. If it fails to read from file it returns
240   * false.
241   * Fulfills task #2 from assignment.
242   */
243  bool ReadTransactionRecord(ifstream &infile, transactionT &t) {
244      string dollars;
245      char c;
246
247      if (infile >> t.customer_number) {
248          infile.get(c);  // get rid of singe space
249          getline(infile, t.memo, '$');
250
251          getline(infile, dollars, '.');
252          infile >> t.amount;
253          t.amount += stoi(dollars) * 100;
254
255          return true;
256      }
257      return false;
258  }
259
260  /* Function: WriteBalanceRecord
261   * Usage: WriteBalanceRecord(balanceout, balance);
262   * ----------------------------------------------
263   * Takes a balance record and writes it to a file that stores balance records.
264   * Fulfills task #6 from assignment.
265   */
266  void WriteBalanceRecord(ofstream &outfile, balanceT b) {
267      outfile << left << setw(5) << b.customer_number
268              << setw(15) << b.customer_name
269              << "$ ";
270      WriteInDollars(outfile, b.amount);
271  }
272
273  /* Function: WriteStatementTransaction
274   * Usage: WriteStatementTransaction(statementfile, trans);
275   * -------------------------------------------------------
276   * Takes a transaction record and writes it to a statement file, placing the
```

```
277   * amount in the appropriate column depending on whether it's a withdrawal
278   * or a deposit.
279   * Fulfills the intention of task #3 on the assignment.
280   */
281  void WriteStatementTransaction(ofstream &outfile, transactionT t) {
282      int col_width = 4, tail_width = 12;
283      if (t.amount < 0) {     // write in separate col if withdrawal
284          col_width += tail_width;
285          tail_width = 0;
286      }
287
288      outfile << left << ' '
289              << setw(18) << t.memo
290              << right << setw(col_width) << abs(t.amount / 100) << '.'
291              << setfill('0') << setw(2) << abs(t.amount % 100)
292              << setfill(' ') << setw(tail_width) << "";
293  }
294
295  /* Function: ReadBalanceRecord
296   * Usage: while (ReadBalanceRecord(balancefile, b) { do something with b; }
297   * Side Effects: value of balance struct pointed to in arguments is changed.
298   * ---------------------
299   * This function read balance records from a file and assigns the values to
300   * the proper attributes of a struct. If it fails to read from file it returns
301   * false.
302   * Fulfills task #2 on assignment.
303   */
304  bool ReadBalanceRecord(ifstream &infile, balanceT &b) {
305      string dollars;
306      char c;
307
308      if (infile >> b.customer_number) {
309          infile.get(c);    // get rid of single space
310          getline(infile, b.customer_name, '$'); // leaves trailing whitespace
311
312          getline(infile, dollars, '.');
313          infile >> b.amount;
314          b.amount += stoi(dollars) * 100;
315
316          return true;
317      }
318      return false;
319  }
320
321  /* Function: DisplayBalanceRecord
322   * Usage: DisplayBalanceRecord(balance);
323   * -------------------------------------
324   * This function takes a balance record in the form of a struct and displays
325   * it in a row via cout.
326   * Fulfills part of task #5 from assignment.
327   */
328  void DisplayBalanceRecord(balanceT b) {
329      cout << left << setw(5) << b.customer_number
330           << setw(15) << b.customer_name
331           << "$ ";
332      WriteInDollars(cout, b.amount);
333  }
```