```
*****************************************************
*   PROGRAMMED BY : Carlos Aguilera
*   CLASS         : CS1B
*   SECTION       : MW: 7:30p - 9:50p
*   LAB #1        : Functions & Arrays
*****************************************************

What input file would you like to use? inFile.txt
What output file would you like to use? oFile.txt

Menu Options

1 - Find the larger balance
2 - Find the smaller balance
3 - Obtain the sum of all balances
4 - Obtain the average of all balances
5 - Find Person
0 - Exit
Enter an option (0 to exit): 1

Finding the Larger Balance...

Menu Options

1 - Find the larger balance
2 - Find the smaller balance
3 - Obtain the sum of all balances
4 - Obtain the average of all balances
5 - Find Person
0 - Exit
Enter an option (0 to exit): 2

Finding the Smaller Balance...

Menu Options

1 - Find the larger balance
2 - Find the smaller balance
3 - Obtain the sum of all balances
4 - Obtain the average of all balances
5 - Find Person
0 - Exit
Enter an option (0 to exit): 3

Obtaining the sum of all Balances...

Menu Options

1 - Find the larger balance
2 - Find the smaller balance
3 - Obtain the sum of all balances
4 - Obtain the average of all balances
5 - Find Person
0 - Exit
Enter an option (0 to exit): 4

Obtaining the average of all Balances...

Menu Options
```

```
1 - Find the larger balance
2 - Find the smaller balance
3 - Obtain the sum of all balances
4 - Obtain the average of all balances
5 - Find Person
0 - Exit
Enter an option (0 to exit): 5

Who do you want to search for (enter done to exit): Steve Woolston
Found.
Who do you want to search for (enter done to exit): Jacques Rousseau
Jacques Rousseau was not found.
Who do you want to search for (enter done to exit): Chris Carroll
Found.
Who do you want to search for (enter done to exit): Lisa Covi
Found.
Who do you want to search for (enter done to exit): Florence Rousseau
Found.
Who do you want to search for (enter done to exit): Frankie Lane
Frankie Lane was not found.
Who do you want to search for (enter done to exit): done

Menu Options

1 - Find the larger balance
2 - Find the smaller balance
3 - Obtain the sum of all balances
4 - Obtain the average of all balances
5 - Find Person
0 - Exit
Enter an option (0 to exit): 0

Thank you for using my program.
```

```
Larger Balance:
ID #     NAME                    BALANCE DUE
----     --------------------    -----------
1002     Steve Woolston          $    1423.2

Smaller Balance:
ID #     NAME                    BALANCE DUE
----     --------------------    -----------
1003     Don McBride             $     12.32

Sum of Balance for all persons:
$   4080.53

Average Balance for all persons:
$    408.05

Search Balance:
ID #     NAME                    BALANCE DUE
----     --------------------    -----------
1002     Steve Woolston          $   1423.20
1008     Chris Carroll           $     32.35
1009     Lisa Covi               $    332.35
1010     Florence Rousseau       $   1323.38

```

```
#pragma once

#include <iostream>
#include <iomanip>
#include <string>
#include <fstream>

struct Account {
    std::string userName;
    int userID;
    double userBalance;
};

void heading();
void readFile(std::string inputFileName, size_t sizeofArray, Account arrayofAccounts[]);//reads file and sets values in the arrays
int balanceIndex(char selection, size_t sizeofArray, Account arrayofAccounts[]);//returns index of largest or smallest balance depending on choice
double sumofBalances(size_t sizeofArray, Account arrayofAccounts[]);//sums all balances in array
int searchName(std::string inputName, size_t sizeofArray, Account arrayofAccounts[]);//searches name specified from user
void handleOutput(int index, std::string type, std::fstream &outFile, const char selection, const size_t sizeofArray, Account arrayofAccounts[]);//handles balance output
void handleSearchOutput(const std::string inputName, const int index, std::fstream &outFile, Account arrayofAccounts[]);//handles search output
```

```cpp
/*****************************************************************************
*
 * AUTHOR      : Carlos Aguilera
 * STUDENT ID  : 1152562
 * LAB #       : 1
 * CLASS       : CS1B
 * SECTION     : M-W
 * DUE DATE    : 02.02.22
 *****************************************************************************
/

#include "main.hpp"
/*****************************************************************************
 * Title: Functions & Arrays
 * ---------------------------------------------------------------------------
 * FUNCTION:
 *  handles output of searches
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 * char selection IN & CALC - char for switch case condition
 * std::string inputFileName IN & CALC - file name for input
 * std::string outputFileName IN & CALC - file name for output
 * std::string inputName IN & CALC - name input to search for from user
 * std::fstream inFile CALC - read from input file
 * std::fstream outFile CALC - output to file
 * size_t sizeofArray CALC - size of the array
 * Account arrayofAccounts[] CALC - array of Accounts
 *****************************************************************************/

int main()
{
  heading();

  std::string inputFileName {}, outputFileName {}, temp {};
  std::cout << "What input file would you like to use? ";
  std::cin >> inputFileName; //reads input for what file to read from
  std::cout << "What output file would you like to use? ";
  std::cin >> outputFileName; // reads input for what file to write to

  size_t sizeofArray {0};
  std::fstream inFile;
  inFile.open(inputFileName, std::ios::in); //file is in read only mode
  while(std::getline(inFile, temp))//stores line in temporary string
    ++sizeofArray;// a loop that gets the number of lines in the file
  sizeofArray /= 2;// works for the type of formatting that the input file
has if format changes then bugs could occur
  inFile.close();

  Account arrayofAccounts[sizeofArray];

  readFile(inputFileName, sizeofArray, arrayofAccounts);//reads file and sets
values in the arrays

  char selection {};
  std::fstream outFile;
  outFile.open(outputFileName, std::ios::app);//appends to file and doesn't
erase but adds instead
  do
```

```
55    {
56      std::cout << "\nMenu Options\n\n"
57          << "1 - Find the larger balance\n"
58          << "2 - Find the smaller balance\n"
59          << "3 - Obtain the sum of all balances\n"
60          << "4 - Obtain the average of all balances\n"
61          << "5 - Find Person\n"
62          << "0 - Exit\n"
63              << "Enter an option (0 to exit): ";
64      std::cin >> selection;
65      std::cout << "\n";
66
67      switch (selection)
68      {
69        case '1': {
70          std::cout << "Finding the Larger Balance...\n";
71          std::string type {"Larger"};
72          int index {balanceIndex(selection, sizeofArray, arrayofAccounts)};
73          handleOutput(index, type, outFile, selection, sizeofArray,
arrayofAccounts);
74          break;
75        }
76        case '2': {
77          std::cout << "Finding the Smaller Balance...\n";
78          std::string type {"Smaller"};
79          int index {balanceIndex(selection, sizeofArray, arrayofAccounts)};
80          handleOutput(index, type, outFile, selection, sizeofArray,
arrayofAccounts);
81          break;
82        }
83        case '3': {
84          std::cout << "Obtaining the sum of all Balances...\n";
85
86          outFile << "Sum of Balance for all persons:\n";
87          outFile << std::fixed << std::setprecision(2) << "$" << std::setw(10)
<< sumofBalances(sizeofArray, arrayofAccounts) << "\n\n";//returns sum of
balances
88          break;
89        }
90        case '4': {
91          std::cout << "Obtaining the average of all Balances...\n";
92
93          outFile << "Average Balance for all persons:\n";
94          outFile << std::fixed << std::setprecision(2) << "$" << std::setw(10)
<< sumofBalances(sizeofArray, arrayofAccounts)/sizeofArray <<
"\n\n";//returns average of balances using the size of array or how many IDs
we have
95          break;
96        }
97        case '5': {
98          std::string inputName {};
99          int index {};
100         std::cin.ignore(10, '\n');
101         do
102         {
103           std::cout << "Who do you want to search for (enter done to exit):
";
104           std::getline(std::cin, inputName);
105
```

```
106           if(inputName == "done") { //expection handling for when user enters
      done
107             handleSearchOutput(inputName, index, outFile, arrayofAccounts);
108             continue;
109           }
110           else if((index = searchName(inputName, sizeofArray,
      arrayofAccounts)) != -1) {// function returns a -1 if not found and if found
      returns index that it was found in
111             std::cout << "Found.\n";
112             handleSearchOutput(inputName, index, outFile, arrayofAccounts);
113           }else
114             std::cout << inputName << " was not found.\n";//handling not
      found
115         } while (inputName != "done");
116
117         break;
118       }
119       case '0': {
120         std::cout << "Thank you for using my program.\n";
121         break;
122       }
123       default: {
124         std::cout << "Invalid input!\n";
125         break;
126       }
127     }
128   } while (selection != '0');
129   outFile.close();
130   return 0;
131 }
```

```cpp
#include "main.hpp"
/*****************************************************************************
 * Title: readFile
 * ---------------------------------------------------------------------------
 * FUNCTION:
 *  Reads from file and stores data in an array of accounts
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 * std::string fname, lname CALC - reads first name and last name
 *****************************************************************************/

void readFile(std::string inputFileName, size_t sizeofArray, Account arrayofAccounts[])//reads file and sets values in the arrays
{
  std::string fname{}, lname{};
  std::fstream inFile;
  inFile.open(inputFileName, std::ios::in); // read mode
  for (size_t i{0}; i < sizeofArray; i++)
  {
    inFile >> fname >> lname;            // wanted to use getline but then thought this could be less error prone
    arrayofAccounts[i].userName = fname + ' ' + lname;       // concat fname and lname and assigning it to array names at index i
    inFile >> arrayofAccounts[i].userID >> arrayofAccounts[i].userBalance; // reading id and balance
  }
  inFile.close();
}
```

```cpp
#include "main.hpp"
/*****************************************************************************
 * Title: balanceIndex
 * ---------------------------------------------------------------------------
 * FUNCTION:
 *  Handles both larger and smaller balances and returns the index that it
 *  found
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 * double balance CALC - used to calc current largest or smallest balance
 *****************************************************************************/

int balanceIndex(char selection, size_t sizeofArray, Account
arrayofAccounts[])
{
  int index {};
  double balance {arrayofAccounts[0].userBalance};//did not want to use a
nested for loop so initialized balance to index 0 of arrayofBalances
  for(size_t i {1}; i < sizeofArray; i++)
  {
    if(selection == '1' && arrayofAccounts[i].userBalance > balance)//if array
of balances at index i is greater than balance(because we want the largest)
then assign it to balance and assign i to index
    {
      balance = arrayofAccounts[i].userBalance;
      index = i;
    }else if(selection == '2' && arrayofAccounts[i].userBalance < balance) {
      balance = arrayofAccounts[i].userBalance;
      index = i;
    }
  }
  return index;
}
```

```cpp
#include "main.hpp"
/*******************************************************************************
 * Title: handleOutput
 * ------------------------------------------------------------------------------
 * FUNCTION:
 *  handles output of balances
 * ------------------------------------------------------------------------------
 * No Data Table
 * ----------
 *******************************************************************************/

void handleOutput(int index, std::string type, std::fstream &outFile, const char selection, const size_t sizeofArray, Account arrayofAccounts[])
{
    outFile << type << " Balance:\n";
    outFile << "ID #     NAME                    BALANCE DUE\n";
    outFile << "----     --------------------    -----------\n";
    outFile << arrayofAccounts[index].userID << "     ";//returns an index for the largest balance in the input file
    outFile << arrayofAccounts[index].userName;
    outFile << std::setw(26 - arrayofAccounts[index].userName.size());//returns the size of the largest balance name and subtracts a set width of 26 to get proper format
    outFile << "$" << std::setw(10) << arrayofAccounts[index].userBalance << "\n\n";

}
```

```cpp
#include "main.hpp"
/*****************************************************************************
 * Title: sumofBalances
 * ---------------------------------------------------------------------------
 * FUNCTION:
 *  sums all balances in the array and returns it
 * ---------------------------------------------------------------------------
 * Data Table
 * ----------
 * double balanceSum CALC - calcs balances sum
 *****************************************************************************/

double sumofBalances(size_t sizeofArray, Account arrayofAccounts[])
{
  double balanceSum {};
  for(size_t i {0}; i < sizeofArray; i++)
    balanceSum += arrayofAccounts[i].userBalance;//takes in balance for each iteration of the for loop and plus equals it
  return balanceSum;//returns result
}
```

```cpp
#include "main.hpp"
/*****************************************************************************
 * Title: searchName
 * ---------------------------------------------------------------------------
 * FUNCTION:
 *  handles name search from user and if name was not found returns -1
 * ---------------------------------------------------------------------------
 * No Data Table
 * ----------
 *****************************************************************************/

int searchName(std::string inputName, size_t sizeofArray, Account arrayofAccounts[])
{
    for(size_t i {0}; i < sizeofArray; i++)
    {
        if(inputName == arrayofAccounts[i].userName)//checking if the name user inputed exists
            return i;// returns index if true
    }
    return -1;//returns -1 if not found
}
```

```cpp
#include "main.hpp"
/*******************************************************************************
 * Title: handleSearchOutput
 * -----------------------------------------------------------------------------
 * FUNCTION:
 *  handles output of searches
 * -----------------------------------------------------------------------------
 * Data Table
 * ----------
 * static int displayCounter CALC - used to determine when to run header
 ******************************************************************************/

void handleSearchOutput(const std::string inputName, const int index,
std::fstream &outFile, Account arrayofAccounts[])
{
    static int displayCounter {0};
    if (inputName != "done") {
        if(displayCounter < 1) {
            //start of header
            outFile << "Search Balance:\n";
            outFile << "ID #     NAME                      BALANCE DUE\n";
            outFile << "----     --------------------      -----------\n";
            ++displayCounter;
            //end of header
        }
        outFile << arrayofAccounts[index].userID << "     ";//returns an index
for the largest balance in the input file
        outFile << arrayofAccounts[index].userName;
        outFile << std::setw(26 -
arrayofAccounts[index].userName.size());//returns the size of the largest
balance name and subtracts a set width of 26 to get proper format
        outFile << "$" << std::setw(10) << arrayofAccounts[index].userBalance
<< "\n";
    }else
        displayCounter = 0;//reset display counter good for reusability
}
```

```cpp
#include "main.hpp"

void heading()
{
    /*****************************************************************************
     * CONSTANTS
     * ---------------------------------------------------------------------------
     * OUTPUT - USED FOR CLASS HEADING
     * ---------------------------------------------------------------------------
     * PROGRAMMER : Programmer's Name
     * CLASS      : Student's Course
     * SECTION    : Class Days and Times
     * LAB_NUM    : Lab Number (specific to this lab)
     * LAB_NAME   : Title of the Lab
     *****************************************************************************/
  const char PROGRAMMER[] = "Carlos Aguilera";
  const char CLASS[]      = "CS1B";
  const char SECTION[]    = "MW: 7:30p - 9:50p";
  const int LAB_NUM       = 1;
  const char LAB_NAME[]   = "Functions & Arrays";

  /*****************************************************************************
   * OUTPUT - Class Heading
   *****************************************************************************/
  std::cout << std::left;
  std::cout << "*************************************************\n";
  std::cout << "*   PROGRAMMED BY : " << PROGRAMMER << std::endl;
  std::cout << "*   " << std::setw(14) <<"CLASS" << ": " << CLASS <<
std::endl;
  std::cout << "*   " << std::setw(14) <<"SECTION" << ": " << SECTION <<
std::endl;
  std::cout << "*   LAB #" << std::setw(9) << LAB_NUM << ": " << LAB_NAME <<
std::endl;
  std::cout << "*************************************************\n\n";
  std::cout << std::right;
  /*****************************************************************************/
}
```

```
Jean Rousseau
1001 15.50
Steve Woolston
1002 1423.20
Michele Rousseau
1005 52.75
Pete McBride
1007 500.32
Florence Rousseau
1010 1323.38
Lisa Covi
1009 332.35
Don McBride
1003 12.32
Chris Carroll
1008 32.35
Yolanda Agredano
1004 356.00
Sally Sleeper
1006 32.36
```