**MEAN AND MEDIAN OF A FILE - LAB REPORT**

**1) Enter your name, student ID, platform (Mac or PC) and date**

Name and Student ID: Samuel Indurkar, 0888068

Class: CIS054 C/C++ Programming

Platform (Mac or PC): gcc & eclipse on MAC  
Date: 7/6/2017

**DESCRIPTION:**In this lab assignment, you are to read a file with an odd number of records, compute the mean and median of the field that has account balances. Repeat the process with a file that has an even number of records. The two files are provided as part of the lab assignment and are named Balances-1.txt and Balances-2.txt. You need to run the program for each file and compute the median value for each file.

NOTE: The code for processing a disk file is very different between the C and C++ languages. There are separate code examples for C and C++. The code samples show how to read a disk file and skip records until you reach the location of the records that will contain the values to be used when computing the median value. You can verify that you have the correct results by opening the files with a spreadsheet program (Microsoft Excel, Apple Numbers or OpenOffice Calc), select the column with the balances and compute the median value with one of the statistical functions.

Sample code is given at the end of this lab assignment. Replace the cout that is inside the while loop with code to compute the total of all balances. After the while loop completes, display the total of all balances.

To open a file, you must specify the full pathname and filename. This is called the fully-qualified filename. For example, when in C or C++, to open a file named Balances.txt that is located on the desktop, the fully-qualified filename on my system would be:

**/Users/Dan/Desktop/Balances.txt**For this lab, you will need to process the **Balances-1.txt** and **Balances-2.txt**

The pathname consists of your HOME directory and /Desktop which is a folder on the operating system that refers to the desktop. For example, my HOME directory is located at **/Users/Dan**

My desktop is located at **/Users/Dan/Desktop** This is true for either a Mac or PC.

**Mac** - To display the location of the HOME directory on a Mac running OS/X

1) Go to **Launchpad** and launch **Terminal**

2) at the Terminal prompt, type **echo $HOME**

3) use this information followed by **/Desktop/Balances**.txt as the fully-qualified filename

**Windows** - To display the location of the HOME directory on a PC running Windows

1) click the START button on Windows located at the lower-left side of the screen

2) when the box pops up at the bottom that says, "Search programs and files", type cmd and press [Enter]

3) at the command prompt, type: **echo %UserProfile%**

4) use this information followed by /Desktop/Balances.txt as the fully-qualified filename

SPECIAL NOTE FOR PROGRAMMERS ON WINDOWS SYSTEMS

Windows uses the backslash character \ to separate parts of the pathname and filename. The C programming language was created to create the Unix operating system which uses the forward slash character / to separate the parts of the pathname and filename.

C and C++ use the backslash \ in character strings as an escape to give a new meaning to the next character. For example, \n is new line, \b is backspace, \t is tab, etc. It is necessary to type two backslashes \\ inside of a quoted string to get one backslash \ in memory. For example to get this:

**\Users\Dan\Desktop**

The quoted string in a C or C++ program must be written as:

**"\\Users\\Dan\\Desktop"**

Since web servers were originally developed on Unix systems, the forward slash / is used in web addresses. To make Windows compatible with the rest of the world, Windows now accepts forward slashes / as well as backslashes \ as part of a filename and they can be used interchangeable. In C or C++ you can now use

**"/Users/Dan/Desktop"**

as well as

**"\\Users\\Dan\\Desktop"**

to get the same result when referring to a file.

**LAB REPORT:  
2) Determine the Inputs, Processing and Outputs before creating the program.**

|  |  |  |
| --- | --- | --- |
| **INPUTS** | **PROCESSING** | **OUTPUTS** |
| filename  Balance1.txt and Balance2.txt | Open the file.  while not end of file  read acctBalance and accumulate; Also count number of lines.  compute mean = accumulatedBalance / totalRecordCount  close file and open file again  skip half of the file  based on odd/even recalculate median | display mean and median |

**3) Fill in the EXPECTED & ACTUAL RESULTS.**

|  |  |  |
| --- | --- | --- |
| **TEST DATA VALUES** | **EXPECTED RESULT**  Use a spreadsheet program to determine the expected median values | **ACTUAL RESULT**  Fill in the output displayed  by the program |
| Balances-1.txt  Mean  Median  Balances-2.txt  Mean  Median | Balance1.txt  The mean is 1136.13  Balance2.txt  The mean is 1132.46 | Balance1.txt  There are 117 records in Balances1.txt  The mean is 1136.13  The median of Balances1.txt is 1149.42  Balance2.txt  There are 118 records in Balances2.txt  The mean is 1132.46  The median of Balances2.txt is 1147.81 |

**DISCUSSION:**

**4) Complete the DISCUSSION section. It does not need to be long, but it needs to be complete.**4a) What did you do to develop the program? ("Followed the Directions" is not a complete description)

Prompt the user for filename.

Open the file.

then check for open error, if error during open, then exit, else proceed.

while not end of file

read acctBalance and accumulate;

Also count number of lines.

compute mean = accumulatedBalance / totalRecordCount

close file and open file again

skip half of the file

based on odd/even count, determine median

Print mean and median

4b) What problems did you have and how did you overcome the problems?

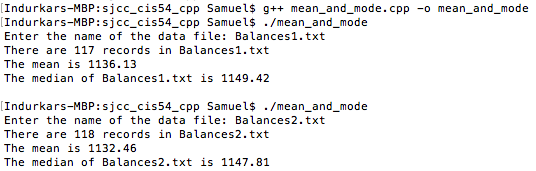
no major problems during coding. A minor hurdle, Initially I did not create Balance1.txt and executed the program, so it said “Unable to open —Balances1.txt—" Then I went back to your instructions and figured it out.

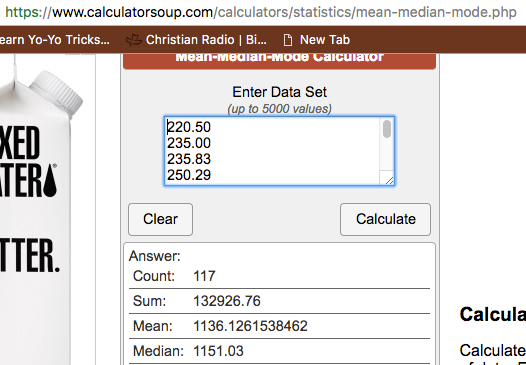
However, a major problem was using excel. I dont have windows. Your video describes the procedure to use excel on windows. I don’t know how to do that on MAC. I just went on internet and tried using google-spreadsheet, but it wasn’t able to read Balance1.txt from my desktop properly in columns and rows. So I abandoned that idea and found some websites which do that. I have attached the screenshot of those.

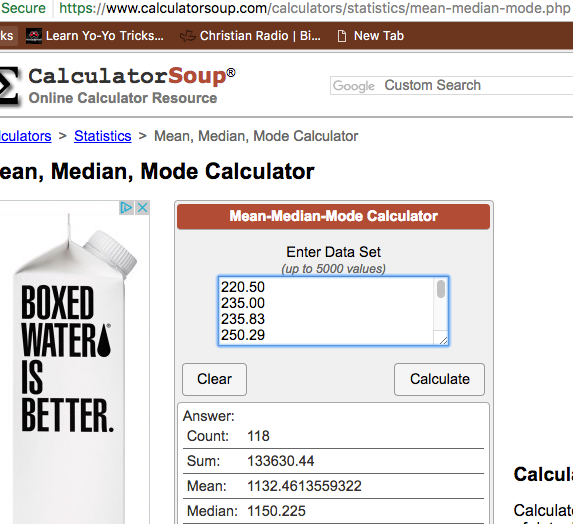
**PROGRAM OUTPUT:**

**5) Display a total of four screenshots. Show two screen shots for each data file. Show a screenshot displaying the result of using a spreadsheet program to calculate the median and mean, and a screenshot showing the execution of the program.**

Refer to previous lab assignments for instructions on how to capture a screen or portions of a screen for either the PC or a Mac

****

****

****

**PROGRAM LISTING:**

**6) Copy and paste the code that YOU typed to make the program work. Your program should include a comment block at the top that shows the name of the program, date, version and your name.**

**/\***

**\* mean\_and\_mode.cpp**

**\***

**\* Created on: Jul 6, 2017**

**\* Author: Samuel**

**\*/**

**// MeanAndMedianOfDataFile.cpp**

**//**

**//**

**#include <iostream>**

**#include <fstream>**

**using namespace std;**

**int main(int argc, char\* argv[])**

**{**

**// variables to control the disk file**

**ifstream infile;**

**char filename[200];**

**int recordCount = 0;**

**int recordsToSkip = 0;**

**int is\_odd;**

**// variables for fields of each record in the file**

**int AcctNo = 0;**

**char Name[100] = "";**

**double AcctBal = 0.0;**

**// varible used to determine the median**

**double totalAcctBal;**

**double mean;**

**double median = 0.0;**

**cout << "Enter the name of the data file: ";**

**cin >> filename;**

**// ---- open the file**

**infile.open(filename);**

**//check if open was succesful**

**if (infile.fail())**

**{**

**cerr << "Unable to open the file first time: " << filename << endl;**

**return(1);**

**}**

**// ---- PART 1, Count the number of records in the file**

**while (!infile.eof()) // while not end of file**

**{**

**Name[0] = 0; // initialize to 0 to test for empty records/**

**infile >> AcctNo >> Name >> AcctBal;**

**if (Name[0] != 0) // ignore empty records**

**{**

**recordCount++;**

**totalAcctBal += AcctBal;**

**}**

**}**

**mean = totalAcctBal/recordCount;**

**// close the file**

**infile.close();**

**cout << "There are " << recordCount << " records in " << filename << endl;**

**cout << "The mean is " << mean << endl;**

**// ---- PART 2, Determine the number of records to skip**

**if (recordCount %2 == 1)**

**{**

**is\_odd = 1;**

**recordsToSkip = recordCount/2; // Odd number of records**

**}**

**else**

**{**

**is\_odd = 0;**

**recordsToSkip = recordCount/2 - 1; // Even number of records**

**}**

**// ---- PART 3, open the file, skip leading records, determine the mean**

**infile.open(filename);**

**//check if open was succesful**

**if (infile.fail())**

**{**

**cerr << "Unable to open the file second time: " << filename << endl;**

**return(1);**

**}**

**while (recordsToSkip != 0)**

**{**

**infile >> AcctNo >> Name >> AcctBal;**

**recordsToSkip--;**

**}**

**if (is\_odd == 1)**

**{**

**// odd**

**median = AcctBal;**

**}**

**else**

**{**

**//even**

**median = AcctBal;**

**infile >> AcctNo >> Name >> AcctBal;**

**median = (median + AcctBal) /2;**

**}**

**infile.close();**

**// Display the results**

**cout << "The median of " << filename << " is " << median << endl << endl;**

**return 0;**

**}**