

Simple Algebraic Calculations

One of the first things you will learn about C++ is how to perform numerical computations. In this project, you are given an incomplete program (see the end of this specification), which already includes the statements necessary to read input and specify output. You must add statements so that your completed program will perform the specified calculations.

The input file:

The input file for this program is named "TripData.txt". A sample input file is given below. The input file begins with two header lines which are meaningful to a human reader but must be ignored by the program. Lines of data, each pertaining to a particular trip that has been taken, follow the header section. Each of these lines specifies the name of the place where the trip began, the name of the destination where the trip ended, the number of miles traveled, and the time required for the trip.

Origin	Destination	Miles	Time
Blacksburg, VA	Knoxville, TN	244	3:25
Knoxville, TN	Nashville, TN	178	2:40
Nashville, TN	Memphis, TN	210	3:20
Memphis, TN	Little Rock, AR	137	2:05
Little Rock, AR	Texarkana, TX	141	2:10
Texarkana, TX	Dallas, TX	180	2:50
Dallas, TX	Ft Worth, TX	30	0:40
Ft Worth, TX	Abilene, TX	155	2:20
Abilene, TX	Pecos, TX	242	3:45
Pecos, TX	Carslbad, NM	75	1:10

The names are strings of characters, which may contain spaces; in order to make the data easier to read in, a single tab character follows each name. A single tab character also separates the mile and time values, and there is a newline character immediately after the time. Note that the appearance of tabs is somewhat unreliable. In the sample above the tabs are displayed according to the settings in Microsoft Word and everything is neatly aligned. Viewed in a text editor, such as the Visual C++ IDE, a tab is usually interpreted as having a fixed width (such as 4 spaces) and the alignment will not appear to be nearly as nice. The visual appearance of the data doesn't matter to the program that will read it, but the presence of tabs will make it easier to write code to read that data.

There may be data for an arbitrary number of trips; i.e., we cannot make any assumptions about the minimum or maximum number of trips for which we will be given data.

As your class covers input and output in C++, you should examine the given program and make sure you understand precisely how it manages its input and output.

What must be calculated:

For each trip, the program must calculate the average speed in miles per hour, MPH, and the time required for the trip in minutes.

Also, the program must calculate summary data for the complete set of trips. Specifically, the program must calculate the number of trips reported, the total miles traveled, the total time required for all the trips, and the average speed in MPH for all the trips.

Note: the average MPH over all the trips is NOT just the sum of the individual MPHs divided by the number of trips. Think about it...

Warning: the necessary calculations are relatively simple, but they may require some thought. It would be a violation of the Honor Code to explain to another student how to do the calculations or to post that information to a course listserv, discussion board, or newsgroup.

In order to produce the most accurate results possible in C++, all the mileage and time values should be stored as integers, not as decimal numbers, and the average speed should be stored using variables of type `double`, not type `float`.

Although the given program is incomplete, it does declare all the variables that are needed to calculate the required results. You may declare additional variables if you like, but they will not be logically necessary.

The output file:

The output file is named "Summary.txt". An output file, which corresponds to the given input file, is shown below. The first two lines specify the programmer and the assignment, and the third is blank. The fourth line specifies column headers for the table the program will write, and the fifth line separates the table data from the labels.

The table body contains one line of output for each given trip, displaying the trip origin and destination, the trip mileage, the length of the trip in minutes, and average speed for that trip.

When a decimal number is written, the number of digits displayed after the decimal point is called the precision of the displayed value. The average speed for each must be written with precision 1, but the average speed over all the trips must be written with precision 2 (as shown in the output sample). The total time for all the trips must be printed in HH:MM format (as shown in the input and output samples).

```
Programmer: Bill McQuain
CS 1044 Summer I 2001

Origin          Destination          Mileage  Minutes  MPH
-----
Blacksburg, VA  Knoxville, TN        244      205     71.4
Knoxville, TN   Nashville, TN        178      160     66.8
Nashville, TN   Memphis, TN         210      200     63.0
Memphis, TN     Little Rock, AR     137      125     65.8
Little Rock, AR  Texarkana, TX        141      130     65.1
Texarkana, TX   Dallas, TX           180      170     63.5
Dallas, TX      Ft Worth, TX         30        40     45.0
Ft Worth, TX    Abilene, TX         155      140     66.4
Abilene, TX     Pecos, TX           242      225     64.5
Pecos, TX       Carlsbad, NM         75        70     64.3
-----

Number of trips:      10
Total miles:          1592
Total time:           24:25
Average MPH:          65.20
```

If you have read the description of how the Curator scores your program in the *Student Guide*, you know that is important that you use the same spelling and capitalization for all the labels shown above. The horizontal spacing does not effect scoring unless you combine things that should be separate or separate things that should be combined. In any case, the given code will exactly match the labels and spacing shown above.

Evaluation:

Everything that was said in the specification for Project 1 about testing still applies here. Do not waste submissions to the Curator in testing your program! There is no point in submitting your program until you have verified that it produces correct results on the sample data files that are provided. If you waste all of your submissions because you have not tested your program adequately then you will receive a low score on this assignment. You will not be given extra submissions.

Your submitted program will be assigned a score based upon the runtime testing performed by the Curator System. We will not be evaluating your submission of this program for documentation style. However, you should examine the given program as a guide to acceptable documentation, and include similar comments for the statements you add to it.

For a number of the later projects, we will also evaluate your submission for documentation, and for other requirements. It is best to begin preparing for that now.

Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* can be found at: <http://ei.cs.vt.edu/~eags/Curator.html>

The submission client can be found at: <http://spasm.cs.vt.edu:8080/curator/>

Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:
//
// - I have not discussed the C++ language code in my program with
//   anyone other than my instructor or the teaching assistants
//   assigned to this course.
//
// - I have not used C++ language code obtained from another student,
//   or any other unauthorized source, either modified or unmodified.
//
// - If any C++ language code or documentation used in my program
//   was obtained from another source, such as a text book or course
//   notes, that has been clearly noted with a proper citation in
//   the comments of my program.
//
// - I have not designed this program in such a way as to defeat or
//   interfere with the normal operation of the Curator System.
//
// <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.

The (incomplete) program source code:

```
// Project 2 for CS 1044 Summer I 2001
//
// Student:      <put your name here>
//
// Programmer:   William D McQuain
// OS:          Windows NT Workstation 4.0
// System:       Pentium II 400, 128 MB Memory
// Compiler:     Visual C++ 6.0, Service Pack 4
// Last modified: May 16, 2001
//
// Purpose
// This program reads a list of trip data records, including
// origin, destination, distance and time. For each trip the
// program computes the time in minutes and average MPH. The
// overall time for all trips and overall average MPH are also
// computed.
//
// The program then writes a summary of its findings to an output
// file.
//
#include <iostream> // for cout
#include <fstream>  // for file streams
#include <iomanip>   // for formatting manipulators
#include <string>   // for string variables
#include <climits>  // for INT_MAX
using namespace std; // to put all of the above things in scope

int main() {

    const int MINPERHOUR = 60; // Number of minutes in an hour

    const string dataFileName = "TripData.txt"; // Names of input file...
    const string resultsFileName = "Summary.txt"; // ...and output file.

    ifstream In(dataFileName.c_str()); // Attach an input stream to the
                                        // input file.

    // If the input file does not exist, this will detect that.
    // We handle that by printing an error message and stopping the program.
    if ( In.fail() ) {
        cout << "Data file not found: " // Write an error message...
              << dataFileName           // including the file name.
              << endl                   // Bang "return"
              << "Exiting now..." << endl; // Finish the message.
        return 1; // Terminate a failed execution.
    }

    ofstream Out(resultsFileName.c_str()); // Attach an output stream to the
                                           // output file.
    Out << fixed << showpoint; // Prepare output stream for
                              // decimal output.

    // Print identification information to output file:
    Out << "Programmer: " << "<put your name here>" << endl;
    Out << "CS 1044 Summer I 2001" << endl;
    Out << endl;
```

```

// Print output column headers to output file:
Out << "Origin           Destination           Mileage   Minutes   MPH"
    << endl;
Out << "-----"
    << endl;

string Origin,           // Name of starting point for trip.
        Destination;    // Name of destination for trip.
int     tripMiles,       // Length of trip in miles.
        tripHours,      // Time trip took is given as hh:mm so:
        tripMinutes,    //   hours field
        tripTime;       //   minutes field
int     numTrips        = 0; // Number of trips reported.
int     totalMiles      = 0; // Total length of all trips, in miles.
int     totalMinutes    = 0; // Total time for all trips, in minutes.
double  tripMPH;        // Speed of trip, in miles per hour.

In.ignore(INT_MAX, '\n'); // Skip over the two header lines in the
In.ignore(INT_MAX, '\n'); //   input file.

getline(In, Origin, '\t'); // Read: the name of the trip origin.
getline(In, Destination, '\t'); //   the name of the trip destination.
In >> tripMiles;           //   length of trip in miles
In >> tripHours;          //   hours field for trip time
In.ignore(1, ':');        //   colon separating hours and minutes
In >> tripMinutes;        //   minutes field for trip time

while ( In ) {           // As long as you got new data, continue...

    numTrips++;          // Count this trip.

    //
    //
    // Calculations relevant to the current trip go here.
    //
    //

    Out << Origin;
    Out << setw(20 - Origin.length()) << ' ';
    Out << Destination;
    Out << setw(27 - Destination.length()) << tripMiles
        << setw(10) << tripTime
        << setw(8) << setprecision(1) << tripMPH
        << endl;

    // Skip everything to the beginning of the next input line:
    In.ignore(INT_MAX, '\n');

    // Try to read the data for another trip:
    getline(In, Origin, '\t');
    getline(In, Destination, '\t');
    In >> tripMiles;
    In >> tripHours;
    In.ignore(1, ':');
    In >> tripMinutes;
}

```

```
// Write a delimiter for the bottom of the output table:
Out << "-----"
    << endl;
Out << endl;

// Check to see if there were any trips; otherwise we'll wind up
// dividing by zero later on.
if ( numTrips == 0 ) {
    Out << "No trips were reported." << endl;
    return 0;          // No trips, so stop here.
}

// Print the summary information for all trips:
Out << "Number of trips: " << setw(8) << numTrips << endl;
Out << "Total miles:      " << setw(8) << totalMiles << endl;

double overallMPH = 0.0;    // average speed over all the trips
int    totalHours = 0;      // number of hours for HH:MM display

//
//
// Calculations relevant to the summary for all trips go here.
//
//

// Print the total time and overall MPH for all trips:
Out << "Total time:      " << setw(5) << totalHours;
Out << ":";
Out << setw(2) << setfill('0') << totalMinutes << endl;
Out << setfill(' ');
Out << "Average MPH:      " << setw(8) << setprecision(2)
    << overallMPH << endl;

// Close the input and output files:
In.close();
Out.close();

// Terminate a successful execution:
return 0;
}
```