*C++ requirements*

* The program must make use of a two dimensional array that is 20 rows by 5 columns. The values in the array must be of type double.
* To help facilitate this you need to create a global const int value for the maximum number of columns. *No other global variables are allowed*.
* Your program must properly check for end of file.
* Your program must properly open and close all files.
* You must use function prototypes for all functions (except main).
* You are required to have at least five functions, including main. You need to have a readFile function, an average function, a columnAverage function, and a function to get the smallest value in a row.

The signatures for the readFile, average and columnAverage functions are:

// read input file and populate the array

int readFile(double values[][MAX\_COLUMNS], int maxRows, string inputFileName);

// for the array

double average(double values[][MAX\_COLUMNS], int numberRows);

// for a specified column (column)

double columnAverage(double values[][MAX\_COLUMNS], int column, int numberRows);

There will be unit tests for readFile, average, and columnAverage.

The signature of the function that calculates the smallest value is up to you.

Failure to follow the C++ requirements could reduce the points received from passing the tests.

*General overview*

This program will read the contents of a file into an array and calculate various values based on the contents of the file.

The program will make use of a two dimensional array that has 20 rows and 5 columns. See your *Gaddis* text book from CS 1336 or topic 7.11 in zyBooks for details on two dimensional arrays.

You will be reading in the contents from a file. You will need to read the file name from cin.

The file will consist of up to 20 sets of 5 values. The values will all be double floating point values.

*readFile function*

One function you will be required to have is called readFile. This function will read the input file. Each read will read in 5 columns of information into the next available row in the two dimensional array. The first 5 values are read into row 0, the next 5 values will be read into row 1, and so on up to 20 rows of input. You will need to keep track of how many rows of input you have read in. This could be anywhere from 0 to 20. If there are more than 20 rows of input you should only read in the first 20 rows.

To help facilitate this you need to create a global const int value for the maximum number of columns.

Your program should use thisconst value when creating the arrays or when defining function prototypes and function headers.

You will also be creating a const int value for the maximum number of rows, but this will be in your main function.

You should not be hard coding 5 or 20 in your code anywhere except in the two const int declarations. The only possible exception to this is the readFile function. It is easier to just read in all 5 columns at a time. If you want to just read in an entire row at a time you can do that in the readFile function. It is possible to write your code to use the MAX\_COLUMNS value and read in one value at a time, but it makes the program logic more difficult. Once you have your code working and submitted you might want to see if you can read in the values using MAX\_COLUMNS and reading in just one value at a time. This enhancement is *not* required for the assignment.

No other use of global variables is allowed.

*Here is an example of some partial code:*

...

const int MAX\_COLUMNS = 5;

// read input file and populate the array

int readFile(double values[][MAX\_COLUMNS], int maxRows, string inputFileName);

...

int main()

{

const int MAX\_ROWS = 20;

string inputFileName;

double grades[MAX\_ROWS][MAX\_COLUMNS];

...

int actualRows = readFile(grades, MAX\_ROWS, inputFileName);

...

}

...

Note that in the above code we are not including the max number of rows in the array when we pass it as the parameter on the readFile function. That is done so we could use the readFile function for different arrays with a different number of possible rows.

The readFile function will open the file and read the data into the passed in array. It will also close the file once it has read in the data. If the file cannot be opened the function should return -1 to the calling function. If the file does not contain any data (or contains less than 5 values) it will return 0. Otherwise the function returns the number of row read, up to 20. The file could contain more than 20 rows of data, but you should only read in the first 20. Do not try to read into the array past the valid limits of the array.

*main processing*

In this program you are going to read in the file name from cin. The file name will be passed to the readFile function. If the file exists the input file will contain 0 or more rows. The readFile function will return -1 if the file does not exist and 0 to MAX\_ROWS depending on how many rows of data were actually in the file.

Assume a file name of *badfile.txt*. If the file does not exist you should output the following message from main:

File "badfile.txt" could not be opened

and exit the program.

Assume now that input file *grades.txt* exists but does not contain any valid data. The number of rows returned from readFile is 0. Output the following message from main:

The input file "grades.txt" did not contain any data

and exit the program.

If there are 1 to 20 rows the program should:

1. Calculate and display the average of all of values read into the array
2. Calculate and display the average for every column in the array
3. Find and display the smallest value for every row in the array

You will need to have different functions to:

1. Calculate the average for the array
2. Calculate the average for a specified column
3. Find the smallest value in a specified row

You can have additional functions if you want them, but you must have main, readFile and the 3 listed above.

The signatures for the average and columnAverage functions are:

// for the array

double average(double values[][MAX\_COLUMNS], int numberRows);

// for a specified column (column)

double columnAverage(double values[][MAX\_COLUMNS], int column, int numberRows);

There will be unit tests for readFile, average, and columnAverage.

The signatures for other functions are up to you.

See the sample runs for the output requirements. The averages and smallest values must all be written with two digits to the right of the decimal point.

The main function will be the driver. It should create the array, read in the file name, and call all of the other functions, processing the logic as required.

*Sample run 1 (valid data)*

Contents of cin:

grades1.txt

Contents of *grades1.txt*:

61.4 89.5 62.6 89.0 100.0

99.5 82.0 79.0 91.0 72.5

Here is the output to cout:

Processing 2 rows, and 5 columns

Average for all values is 82.65

Average for column 0 is 80.45

Average for column 1 is 85.75

Average for column 2 is 70.80

Average for column 3 is 90.00

Average for column 4 is 86.25

Smallest value for row 0 is 61.40

Smallest value for row 1 is 72.50

*Sample run 2 (invalid file name)*

Contents of cin:

badfile.txt

Here is the output to cout:

File "badfile.txt" could not be opened

*Sample run 3 (value, but empty file)*

Contents of cin:

grades.txt

Here is the output to cout:

The input file "grades.txt" did not contain any data

*Sample run 3 (valid flle that contains more than 20 rows, only read in first 20)*

Contents of cin:

grades2.txt

Contents of *grades2.txt*:

10.0 20.0 30.0 40.0 50.0

100.0 90.0 80.0 70.0 60.0

95.0 85.0 75.0 65.0 55.0

25.2 27.2 29.3 31.2 33.2

88.9 78.8 68.7 58.6 48.5

10.1 20.1 30.1 40.1 50.1

100.1 90.1 80.1 70.1 60.1

95.1 85.1 75.1 65.1 55.1

25.2 27.3 29.4 31.3 33.3

88.0 78.9 68.8 58.7 48.6

10.9 20.8 30.7 40.6 50.5

100.9 90.8 80.7 70.6 60.5

95.9 85.8 75.7 65.6 55.5

25.8 27.7 29.6 31.5 33.4

88.7 78.6 68.5 58.4 48.3

20.0 30.0 40.0 50.0 60.0

90.0 80.0 70.0 60.0 50.0

85.0 75.0 65.0 55.0 45.0

45.2 47.2 49.3 51.2 53.2

48.9 58.8 68.7 78.6 88.5

85.0 75.0 65.0 55.0 45.0

Here is the output to cout:

Processing 20 rows, and 5 columns

Average for all values is 57.21

Average for column 0 is 62.45

Average for column 1 is 59.86

Average for column 2 is 57.24

Average for column 3 is 54.58

Average for column 4 is 51.94

Smallest value for row 0 is 10.00

Smallest value for row 1 is 60.00

Smallest value for row 2 is 55.00

Smallest value for row 3 is 25.20

Smallest value for row 4 is 48.50

Smallest value for row 5 is 10.10

Smallest value for row 6 is 60.10

Smallest value for row 7 is 55.10

Smallest value for row 8 is 25.20

Smallest value for row 9 is 48.60

Smallest value for row 10 is 10.90

Smallest value for row 11 is 60.50

Smallest value for row 12 is 55.50

Smallest value for row 13 is 25.80

Smallest value for row 14 is 48.30

Smallest value for row 15 is 20.00

Smallest value for row 16 is 50.00

Smallest value for row 17 is 45.00

Smallest value for row 18 is 45.20

Smallest value for row 19 is 48.90

Make sure you close your input file and make sure you do not go past the end of the array.

*Expected output*

There are ten tests. For 1, 2, 3 and 8 of the tests you must match, exactly, the expected output. Unit tests 4, 5, 6, and 7 test the readFile function. Unit test 9 tests the average function, and unit test 10 tests the columnAverage function.

You will get yellow highlighted text when you run the tests if your output is not what is expected. This can be because you are not getting the correct result. It could also be because your formatting does not match what is required. The checking that zyBooks does is very exacting and you must match it exactly. More information about what the yellow highlighting means can be found in course "How to use zyBooks" - especially section "1.4 zyLab basics".

Finally, do not include a system("pause"); statement in your program. This will cause your verification steps to fail.

*Note:* that the system("pause"); command runs the pause command on the computer where the program is running. The pause command is a Windows command. Your program will be run on a server in the cloud. The cloud server may be running a different operating system (such as Linux).

*Error message "Could not find main function"*

Now that we are using functions some of the tests are unit tests. In the unit tests the zyBooks environment will call one or more of your functions directly.

To do this it has to find your main function.

Right now zyBooks has a problem with this when your int main() statement has a comment on it.

For example:

If your main looks as follows:

int main() // main function

You will get an error message:

Could not find main function

You need to change your code to:

// main function

int main()

If you do not make this change you will continue to fail the unit tests.