CSCE 240 – Programming Assignment Three

Due: 11:59pm on Thursday, October 12th

Program Purpose

Implement the functions described below.

CountAboveAv – Function that returns the number of elements in a double-subscripted array of doubles that are larger than the mean value in the array.

CountAboveAv takes two arguments: a double-subscripted array of doubles with 10 columns per row, and an integer holding the number of rows in the array.

Consider the array x pictured below:

1.2	8.7	4.1	6.7	7.1	0.7	0.3	9.4	6.4	5.2
2.9	2.4	2.3	2.1	1.9	3.4	0.6	1.8	1.7	2.2
5.7	8.7	2.3	7.2	3.3	2.1	1.6	4.4	5.5	6.6
0.5	3.5	4.1	1.6	2.5	3.9	0.5	1.8	5.6	5.2

The function call *CountAboveAv(x, 4);* should return 17.

SortByCol - Function to sort the rows in a double-subscripted array of doubles by the values in a given column.
SortByCol takes four arguments: a double-subscripted array of doubles with 10 columns per row, an integer holding the number of rows in the array, an integer for the column to sort by, and a bool for whether to sort in ascending order (true = sort ascending, false = sort descending).

Consider the array x pictured below:

1.2	8.7	4.1	6.7	7.1	0.7	0.3	9.4	6.4	5.2
2.9	2.4	2.3	2.1	1.9	3.4	0.6	1.8	1.7	2.2
5.7	8.7	2.3	7.2	3.3	2.1	1.6	4.4	5.5	6.6
0.5	3.5	4.1	1.6	2.5	3.9	0.5	1.8	5.6	5.2

After the function call *SortByCol(x, 4, 6, true);* the array should be organized as shown below.

1.2	8.7	4.1	6.7	7.1	0.7	0.3	9.4	6.4	5.2
0.5	3.5	4.1	1.6	2.5	3.9	0.5	1.8	5.6	5.2
2.9	2.4	2.3	2.1	1.9	3.4	0.6	1.8	1.7	2.2
5.7	8.7	2.3	7.2	3.3	2.1	1.6	4.4	5.5	6.6

SortByRow - Function to sort the columns in a double-subscripted array of doubles by the values in a given row.
SortByRow takes four arguments: a double-subscripted array of doubles with 10 columns per row, an integer holding the number of rows in the array, an integer for the row to sort by, and a bool for whether to sort in ascending order (true = sort ascending, false = sort descending).

Consider the array x pictured below:

1.2	8.7	4.1	6.7	7.1	0.7	0.3	9.4	6.4	5.2
2.9	2.4	2.3	2.1	1.9	3.4	0.6	1.8	1.7	2.2
5.7	8.7	2.3	7.2	3.3	2.1	1.6	4.4	5.5	6.6
0.5	3.5	4.1	1.6	2.5	3.9	0.5	1.8	5.6	5.2

After the function call *SortByRow(x, 4, 1, false);* the array should be organized as shown below.

0.7	1.2	8.7	4.1	5.2	6.7	7.1	9.4	6.4	0.3
3.4	2.9	2.4	2.3	2.2	2.1	1.9	1.8	1.7	0.6
2.1	5.7	8.7	2.3	6.6	7.2	3.3	4.4	5.5	1.6
3.9	0.5	3.5	4.1	5.2	1.6	2.5	1.8	5.6	0.5

MedianInCol - Function that returns the median value in a given column in a double-subscripted array of doubles.
MedianInCol takes three arguments: a double-subscripted array of doubles with 10 columns per row, an integer holding the number of rows in the array, and an integer for the column to examine.

Consider the array x pictured below:

1.2	8.7	4.1	6.7	7.1	0.7	0.3	9.4	6.4	5.2
2.9	2.4	2.3	2.1	1.9	3.4	0.6	1.8	1.7	2.2
5.7	8.7	2.3	7.2	3.3	2.1	1.6	4.4	5.5	6.6
0.5	3.5	4.1	1.6	2.5	3.9	0.5	1.8	5.6	5.2

The function call *MedianInCol(x, 4, 0);* should return 2.05.

Consider the array x pictured below:

1.2	8.7	4.1	6.7	7.1	0.7	0.3	9.4	6.4	5.2
2.9	2.4	2.3	2.1	1.9	3.4	0.6	1.8	1.7	2.2
5.7	8.7	2.3	7.2	3.3	2.1	1.6	4.4	5.5	6.6

The function call *MedianInCol(x, 3, 4);* should return 3.3.

ModeInCol - Function to determine the most frequently occurring value(s) in a given column in a double-subscripted array of doubles.
ModeInCol takes four arguments: a double-subscripted array of doubles with 10 columns per row, an integer holding the number of rows in the array, an integer for the column to examine, and a double array of two elements that will hold the mode(s) when the function exits.
The function will return an integer specifying the number of modes. If a column has more than two values that occur with the same highest frequency, the function will return 0.

Consider the array x pictured below:

1.2	8.7	4.1	6.7	7.1	0.7	0.3	9.4	6.4	5.2
2.9	2.4	2.3	2.1	1.9	3.4	0.6	1.8	1.7	2.2
5.7	8.7	2.3	7.2	3.3	2.1	1.6	4.4	5.5	6.6
0.5	3.5	4.1	1.6	2.5	3.9	0.5	1.8	5.6	5.2

The function call *ModeInCol(x, 4, 3, y);* should return 0 and leave the array *y* unchanged.
The function call *ModeInCol(x, 4, 9, y);* should return 1 and the value in element 0 of array *y* should be 5.2
The function call *ModeInCol(x, 4, 2, y);* should return 2 and the values in array *y* should be 2.3 and 4.1

Additional Specifications

- All function prototypes must be contained in a file named *program3functions.h*
- All function implementations must be written in a file named *program3functions.cc*
- You will submit your *program3functions.h* and *program3functions.cc* files to the assignment in Blackboard
- The only header files that may be included in your code are: iostream, iomanip, cmath, checkArraysMatch.h (in the attached p3files.zip), and program3functions.h. Files that include other headers will not be eligible for correctness points.
- Programs must compile and run on a computer of the instructor's choosing in the Linux lab (see your course syllabus for additional details).
- Be sure to review the program expectations section of the course syllabus.

Initial Testing

Initial tests for the functions are attached to the assignment in Blackboard. A *makefile* has been included to run your functions with the sample tests. In order to use the *makefile,* ensure that your *program3functions.h* and *program3functions.cc files* and all of the files attached to the assignment are in the same directory. Your program will be graded using this same method with modified tests.

The commands to run the sample tests are given below:

```
make testCountAboveAv
make testSortByCol
make testSortByRow
make testMedianInCol
make testModeInCol
```

You are strongly encouraged to create additional, more rigorous tests.

Grade Breakdown

Style: 1 point
Documentation: 1 point
Clean compile of *program3functions.cc*: 1 point
CountAboveAv passes instructor's tests*:* 1 point
SortByCol passes instructor's tests: 1.5 points
SortByRow passes instructor's tests: 1.5 points
MedianInCol passes instructor's tests: 1 point
ModeInCol passes instructor's tests: 2 points

The penalty for late assignment submissions is 10% per day up to three days after the assignment due date. No assignment submissions will be accepted more that 3 days after the due date.