**CSci 1500 - Assignment 5 – 100 pts.**

**Due Date: Nov 19, 2019**

Here are the things you need to do for each problem:

* Design your solutions to produce the program output like that given for each of the problems. Note: Your program should work correctly for any valid user input, not just for the example user input values given.
* Follow the coding guidelines in textbook. Remember to use appropriate data types for all variables. Remember to include each name of your group in a comment at the top of the program. Compile and run the program and verify that it works properly for a variety of input values.

**What you need to turn in:** A printed copy of your C++ code for each of the problems, arranged in order, and stapled together. Include each name of your group on the front page of what you turn in. Clearly identify which C++ code solves which problem.

1. **Arrays/Vectors.** The Fibonacci sequence is a list of numbers in which the first two numbers in the list are both one and each successive number is the sum of the previous two numbers. This means that the first eight Fibonacci numbers are 1, 1, 2, 3, 5, 8, 13, 21. Write a program that will calculate and store the first 20 Fibonacci numbers in an array or vector. Then print a table consisting of each of the first 20 Fibonacci numbers greater than one along with the ratio of that Fibonacci number to its predecessor, shown to 8 digits of accuracy. Here is what the first few lines of output should look like from running your program:

FIB# RATIO

2 2.00000000

3 1.50000000

5 1.66666667

8 1.60000000

13 1.62500000

21 1.61538462

**2. Arrays/Vectors.** Write a program that will prompt for and read in a list of up to ten computer store names and the corresponding price that each store charges for a particular model of PC, and will then determine the lowest price charged for that PC among the stores input. Then, the program should print two lists:

List 1 – The names of all stores charging the lowest price.

List 2 – The names and prices for all stores whose price does not exceed the lowest price by more than 10%.

**Hint:** Use two arrays - one to store the names and the other to store the prices.

**3.Arrays/Vectors of Counting Variables.** A company pays its salespeople on a commission basis. The salespeople receive a weekly take-home salary of $200 per week plus 9% of their gross sales for that week. For example, a salesperson who grosses $5000 in sales in a week receives $200 plus 9% of $5000, or a total of $650. Write a program that will repeatedly prompt for and read in a weekly gross sales amount for each individual salesperson and then calculate and display her take-home salary. In addition, the program should calculate and display the combined total weekly sales for ***all*** of the salespeople, the combined total take-home salary for ***all*** of the salespeople, the total number of salespeople, and a summary of how many of the salespeople earned salaries in each of the following ranges (assume that each salesperson’s weekly salary is truncated to an integer amount):

$200-$299 $300-$399 $400-$499

$500-$599 $600-$699 $700-$799

$800-$899 $900-$999 $1000 and over

Use an array of counters to count these totals. **Note:** Assume that the user of the program does not know how many salespeople there are in advance. To see a sample output from the program, run the program **Assign5\_3.exe** which can be obtained from the **Share** file server: S:\Coursework\Liu\CSCI1500 on campus.

**4.Matrix/2-d Array.** Write a program that reads in the values of a table consisting of three rows and three columns of integers, one row at a time into a 3 x 3 array or matrix and then displays the table followed by the column totals (i.e., the sums of the numbers in each of the three table columns). Here is an example of what output should look like from running your program (user input shown in **bold**):

Enter row 1: **1 2 3**

Enter row 2: **4 5 6**

Enter row 3: **3 2 1**

Table:

1 2 3

4 5 6

3 2 1

------

8 9 10

**5. Using Structures.** Create a structure, **Point**, that will store the coordinates of a point in the *x-y* plane. Then, write a program that will repeatedly prompt for and read in the coordinates of two points (storing them in **Point** variables) and then determine and display the midpoint of the line segment connecting the two points. The program should use a function to find the midpoint – it will receive two points via arguments and will determine and return the midpoint of the line segment connecting the two points. **NOTE:** This function should **return** the midpoint to the calling module, **not display the midpoint**. It should not contain any cout or cin statements. Here is an example of what output should look like from running your program (user input shown in **bold**):

Enter 1st point (x,y): **(1,3)**

Enter 2nd point (x,y): **(2,5)**

Midpoint between (1,3) and (2,5) is (1.5,4)

Continue (y or n)? **y**

Enter 1st point (x,y): **(-2,4)**

Enter 2nd point (x,y): **(-2,0)**

Midpoint between (-2,4) and (-2,0) is (-2,2)

Continue (y or n)? **n**

**Midpoint formula**: The midpoint of the line segment between (*a*,*b*) and (*c*,*d*) is .

**6. Using Structures.** Create a structure, called time, that can be used to store a clock time specified in hours, minutes, and seconds (assume all three of these values will be integers and that military time is being used; i.e., hour values run from 0 to 23). Write a program that will repeatedly prompt for and read two time values and then calculate and display the sum of the two times. Your program should use a function that will accept two times via arguments and will calculate and **return** the sum of the times to the calling module, **not display the sum**. Here is an example of what output should look like from running your program (user input shown in **bold**):

Enter the 1st time (hh:mm:ss): **10:34:55**

Enter the 2nd time (hh:mm:ss): **15:51:15**

Sum of times (hh:mm:ss) = 2:26:10

Enter another time pair (y or n)? **n**