Lab 18: Parallel Array processing **Due:** 11/22/24

In this lab you will practice using parallel array processing to process lists of items.

Your Program

This program must read from an input file called **input18.txt** data as the one shown in Figure 1, process it and send the output to a file called **output18.txt** as shown in Figure 2.

Each row (record) in the input file has a student name followed by her/his actual grade. Your program must find the highest grade of the class and use it to curve the students' grades. To calculate the curved grade, you must use the formula shown below. All grades (Actual and Curved) are whole numbers.

The output must include the data read from the input file, the curved grades, and a graph that uses '*' to implement a horizontal bar chart. Each '*' in the graph represents 10 points of the curved grade (since you cannot have fractions of an asterisk, your program must round off the number of asterisks to be displayed to the nearest integer before they are graphed. Example: 63 => 6 asterisks, 68 => 7 asterisks).

Assume that there can be, at most, data for 40 students in the input file.

$$\text{Curved grade} = \frac{\text{Actual grade}}{\text{Highest grade}} \times 100$$

```
Smith 92
Lopez 65
Garcia 68
Trenton 75
Pitt 90
Garza 95
Tsai 84
Chei 70
Piazza 63
Holtz 58
```

Figure 1

Student Name	Actual Grade	Curved Grade	Graph
Smith	92	97	|**********
Lopez	65	68	|*******
Garcia	68	72	|*******
Trenton	75	79	|********
Pitt	90	95	|**********
Garza	95	100	|**********
Tsai	84	88	|*********
Chei	70	74	|*******
Piazza	63	66	|*******
Holtz	58	61	|******

Figure 2

You **must** define the following functions to process the data:

1) _____ **getData**(_____). Receives the input file, the array of students' names, and the array of students' grades. It reads the input data from the file into the corresponding arrays and returns them along with the quantity of students processed to main(). Although there should be no more than 40 students, this function must ensure that if the file has more records than the maximum capacity of the array, it reads only the first 40 and displays a warning on the screen indicating that the maximum has been exceeded. Use file **input18b.txt** to check this feature in your program.

2) _____ **getMax**(_____). Receives the array of students' grades and the quantity of grades to process. It determines and returns to main() the highest grade of the class.

3) _____ **printData**(_____). Receives the output file, the array of students' names, the array of students' actual grades, the quantity of students to process, and the highest grade of the class. It sends the output to the output file as shown in Figure 2. This function must call a function named **curveGrade()** to get the value for the curved grade (curved grade = actual grade/highest grade*100) of each student.

To create the graph use a **for loop** that displays an '*' per each 10 points of the **curved grade**. You need to round to the nearest integer the quantity of '*' for each student. To do this, use the formula quantity = (curved grade + 5)/10. See examples below:

- for curved grade 61, quantity = (61+5)/10 = 6, so 6 asterisks must be displayed
- for curved grade 66, quantity = (66+5)/10 = 7, so 7 asterisks must be displayed

4) _____ **curveGrade**(_____). Receives the actual grade of a student and the highest grade of the class and returns the curved grade. The arguments that it receives are whole numbers. The result of the division must be rounded off to the nearest integer before it is returned by the function.

Declare a global constant called **MAX** to hold the maximum quantity of records that the program can process.

Function **main()** must just open the input and the output files (ensuring they are opened), call the functions to process the data, and close the files.

Implement the algorithm provided as comments in **lab18.cpp** to create your program. Open it in your IDE and complete the program.

Note:

- Carefully analyze the figures provided above and use them as a reference to ensure you do the right things.
- You must define whether each function is a void or a value-returning function.
- You must define whether each parameter is a value parameter or a reference parameter.
- You must use the most appropriate type of loop and decision statements.
- You must use the most appropriate type of arrays to process the data.

Don't forget to include at the top of the program the comments shown below with your information (name, class and section number, etc.)

```
//////////////////////////////////////////////////////////////////
//
// Name: <Put your name here>
// Date: <Today's date>
// Class: <Your class number and section number, like: CSCI 1470.02>
// Semester: <This semester, like: Fall 2012>
// CSCI 1470 Instructor: <Your lecture instructor's name>
```

//
// Program Description: Enter here **your** description of what the program does
//
//
When done, submit your solution through Blackboard using the "Assignments" tool. Do Not email it.

Paste the link to your final solution along with your source code in the textbox opened when you click on Create Submission before you click on Submit.

The following is the basic criteria to be used to grade your submission:

You start with 100 points and then lose points as you don't do something that is required.

-5: Minor mistakes
incorrect rounding
incorrect function name (each function)
-10: Moderate mistakes
didn't round off
incorrect graph
for not checking going over array capacity
didn't follow the specified format for output.
didn't use value and reference parameters or function's data type correctly (each function)
incorrect function call (each function)
wrong implementation of main()
-20: Major mistakes
incorrect type of array
wrong implementation of a required function (each)
missing graph
didn't use the most appropriate loops where needed
didn't use the most appropriate decision statements where needed
program does not implement the provided algorithm
Incorrect/missing source code or incorrect/missing link to your solution
-30: didn't implement the required functions (each)
-50: didn't use arrays to implement the solution
-40: Program doesn't run correctly
-50: Program doesn't compile
-100: The code submitted is not your creation (you got it from a web site or another person)
-10: Late

Note: more points may be lost for reasons not specified here.