CS2010 Lab 13

### Due: Friday, April 20, 2018



This lab assignment is designed to give you practice using parallel arrays to store and process data. Work with a partner to create a Visual Studio C++ program to implement your solution to the problem. Include both of your names and *lab13* in the project and file name (e.g., **OrtegaPace\_lab13** and **OrtegaPace\_lab13.cpp**). Also include both of your names in the header comments in the file.

Write a complete C++ program to read in and process data to help a company keep track of sales for the eight different types of salsa it makes. The salsa types and jars sold are stored in the file *lab13.txt*. Copy this file from Canvas to your project folder. Your program should declare a constant to hold the array size (8) and define two parallel arrays of that size: an array of strings to hold the salsa names and an array of integers to hold the number of jars sold for each salsa type during the past month. Define the arrays in your *main* function and pass them to the functions that need to use them.

For each task below, add a function prototype before the *main* function, a function call in the *main* function and the function definition after the *main* function. Use a loop in each function to process the arrays. Make sure each function is working before you go on to the next one.

1. Pass the two arrays and the array size to a function that reads the salsa data from the ***lab13.txt*** file and stores it in the arrays. The data file contains two lines for each salsa type: the name of a salsa type is on one line followed by the number of jars sold for that salsa type on the next line. Your function should read in each salsa type and jars sold and store the values in the next available element of the two arrays.

2. Write a second function to display the contents of both arrays in a table like the one below. It should also calculate and display the total number of jars sold (you don't need to return this value). Your output should look something like this:

Ele# Salsa type Jars sold

0 mild 50

1 medium 100

.. ... ...

Total jars sold xxx

3. Write a third function to allow the user to search for a particular kind of salsa. Include a prompt before the function call in the *main* function to ask the user which salsa type they would like to search for. If the salsa type is found in the salsa names array the function should return the element number where it was found. If the salsa type is not found in the array the function should return -1.

Add code in your *main* function after the call to the search function to display the results of the search. For example, if the user types in *zesty* as the type of salsa to search for, the *main* function should display the message:

*There were* ***60*** *jars of* ***zesty*** *salsa sold last month.*

If the user enters *mango* as the salsa type your *main* function should display the message:

*The salsa type* ***mango*** *was not found.*

4. The company would like a list of the salsas showing the types in order from the highest seller to the lowest. Write a fourth function to sort the jars sold array in descending order using the bubble sort algorithm. Be sure to add statements to your bubble sort function so that each time you swap the jars sold values in a pair of elements you also swap the same pair of elements in the salsa types array.

After the sort function is done, the *main* function should call the second function again to display the arrays in sorted order.

5. **Bonus!** (1 pt) The company is always adding new salsa types or removing old ones so the program needs to be modified to handle any number of salsa types up to a maximum of 15 so change the array size constant to 15. Modify your first function so that it counts the number of salsa types actually read in and passes this number back to the *main* function. Store this value in a variable (e.g., *numTypes*). For example, if there were data in the file for ten salsa types, only 10 of the 15 array elements would contain salsa data (a partially-filled array). The number passed back from this function and stored in *numTypes* should be 10.

Modify your other functions to pass the value representing the actual number of elements that contain salsa data (instead of the maximum array size) to each function. Each function should use this value to control how many elements are processed (i.e., if data for 10 types was read in, it should process only the first 10 elements). Test your program by adding (or deleting) 1 or 2 salsa types in the ***lab13.txt*** file. For example, Add *verde* with 25 jars sold and *garlic* with 35 jars sold to the file and test your program. Then delete the salsa type *berry* and its number of jars sold and test your program again.

**6. Turn in your .cpp file only on Canvas when you are done.**

**CS2010 Lab 13**: Grading Rubric

\_\_\_\_\_\_\_ **.cpp file only** turned in on Canvas, named correctly

\_\_\_\_\_\_\_ (1.5)Function correctly reads in salsa data from file and stores it in two parallel arrays. Function prototype, call and definition correct, parameters passed by value/by reference as needed.

\_\_\_\_\_\_\_ (1)Function correctly displays salsa data stored in the arrays and total jars sold. Function prototype, call and definition correct, parameters passed by value/by reference as needed.

\_\_\_\_\_\_\_ (1.5)Function correctly returns value indicating whether a salsa type is found or not, *main* function displays appropriate message.

\_\_\_\_\_\_\_ (1)Function correctly sorts salsa data stored in the arrays in descending order by jars sold. Arrays displayed in sorted order.

\_\_\_\_\_\_\_ (1) **Bonus!** Program correctly handles any number of salsa types up to a

maximum of 15.

**\_\_\_\_\_\_\_** **(5) Total Points** (**6** with bonus)