STL Containers

Complete this worksheet by answering the questions using the given program. Make sure to color or **bold** your answers so they are distinguishable from the question.

Getting Started

- ☐ Make a copy of this document by selecting “Make a Copy” from the File menu.
- ☐ Go to https://replit.com/~ and log in with Gmail using your Weber State email address.
- ☐ Open the code linked below
- ☐ Click **fork**, located in the upper right-hand corner, to create your own copy
- ☐ In your copy, go to the upper right-hand corner, and click **invite**. Then, choose **Generate a Join Link**.
- ☐ Paste the generated link in the second bullet point below

Worksheet Program Links

- *Starter Code*:Repl File: Containers
- *Finished Program Code*: *https://replit.com/join/yfpwionahe-trinanixon1*

Part 1: StudentInfo Observations

In this section, we will observe the setup for the StudentData project. Make observations about the StudentInfo class by answering the following questions.

1. Locate and open the file named student_data.csv. Observe the data within. Look at the first line in the file. Does this line contain data or headers? What does this tell you about the information contained in the file?The first line contains headers. It tells us that the data in the file includes an id, gender, school, and state.
2. What punctuation separates each piece of data? Comma
3. How can you determine when each record ends?each record is on its own line so a new line character would indicate a new piece of data.
4. Observe the private variables in StudentInfo.h? Is there a correlation between the data in the file and the private variables you see in the StudentInfo.h? Yes, they are the same

5. What is the PARSE_CHAR set to? Why do you suppose it is set to that particular character? It's set to ','; The data is a .csv file, which stands for comma-separated values. These types of files are commonly used because a comma is a simple way to separate the data within the string, and they make it easier to parse individual parts of the string.
6. Observe the constructor for the StudentInfo class? What object is used to parse the line from the file? Stringstream ss object is used to parse the line
7. Why do you suppose the PARSE_CHAR is a parameter in the getline() function call? Because then the constructor can use the ',' specified by PARSE_CHAR to split the string and get individual components
8. Why do you suppose PARSE_CHAR is not used in the last item? Because the last item is the state and it is not separated by a comma, so it will just read the rest of the line as the state

Part 2: StudentData Observations

In this section, we will observe the setup for the StudentData project. Make observations about the StudentData class by answering the following questions.

1. Observe the private variable in the StudentData.h file. What type of container will be used to store the StudentData? A vector
2. Observe the Load_Data() function in the StudentData.cpp file. What type of input is reading from the file (>> or getline)? Why do you suppose that particular input is used? It uses getline to read from the file. Getline is a good option when each line of the file represents a single student, and the attributes in the string are separated by commas.
3. Observe the getline statement above the while loop. Why do you suppose the first line is not added to the studentData list? Because it's a header line. It doesn't contain any

actual data, it just describes the data in the lines following it. So the getline(in, line) statement reads and stores the header line in the 'line' variable but doesn't use it.

4. Observe the while loop. What vector function is used to add information to the vector? Push_back is used to add information to the vector. It adds a new element to the end of the vector.

Part 3: Sets

Observe and utilize sets by answering the following questions.

1. Observe the GetStates() function. What container is used in this function? A set<string> container is used.
2. Observe the for loop. Is it looping through the entire list of StudentInfo? Or just a partial list? It's looping through the entire list.
3. Is there anything preventing a state from getting added to the states list? A set<> container only allows unique elements to be added. So in this regard, a state that has already been added will not be added again.
4. Observe the file named student_data.csv. Is the state Maine contained more than one time in the file?Just one time.
5. Run the program. Locate Maine. Is it listed more than once? No, just once.
6. Maine was added to the set more than once, why do you think it is not displaying more than once? This is because a set<> container is being used and they do not allow duplicate elements.
7. What do you think is the purpose of using a set? I would use a set to create a collection of unique elements and/or to remove duplicates efficiently.
8. Locate the GetSchools() function. Complete the code to create a unique set of schools.

9. Locate part 3 in main.cpp. Print out the list of schools by calling the function you just created.
10. Run the program. Your code should look something like this…

```
Schools:
Corporate university
Graduate school
Gymnasium
High school
Junior college
Middle school
Public university
Technical college
University college
University-preparatory school
```

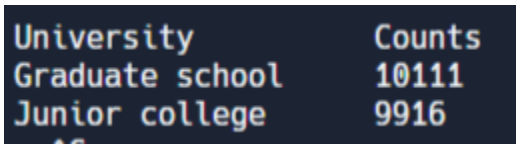
Part 4: Algorithm Lambda's

Observe and utilize algorithms by answering the following questions.

1. Navigate to the main() method in main.cpp. Comment out the function call to Part3() and remove the comment to the function call to Part4().
2. Navigate to the StudentData.cpp file. Observe the CountGender function. What do you suppose the parameter is for? The parameter is string gender, and this is the gender to be counted.
3. What algorithm is used to count the number of students of the given gender? Count_if (for reference: count_if takes 3 arguments: the beginning of the range, the ending of the range, and a condition to be satisfied.)
4. Observe the lambda function used in the count_if algorithm.
    a. What variable is used for the capture clause? gender
    b. What value is used for the parameters of the function? Const StudentInfo &d

c. How is the variable for the capture clause, and the parameter working together to form the return statement for the lambda? It compares the gender of object d with the captured gender using d.getGender() == gender. If they're equal, it returns true so the condition is satisfied for that student.

5. Run the program. How many female vs male students are there in the study? Female: 56999 Male: 43001
6. Navigate to the CountByCollege() function. Complete the code to count all the students attending the given college type.
7. Run the program. You will know it is working if you get the following values.



Part 5: Maps

Observe and utilize maps by answering the following questions.

1. Navigate to the main() method in main.cpp. Comment out the function call to Part4() and remove the comment to the function call to Part5().
2. Navigate to the StudentData.cpp file and observe the GetCountsByUniversity() function. What container is used to get the counts for each university? map
3. What data type is the key? What data type is the value for the map? The data type of the key is string (the keys in the map are strings). The data type for the value is int. So all the values associated with the keys are integers.
4. What container is the loop iterating? studentData_ is the container, which is a collection of "StudentInfo" objects.
5. Observe the if statement? What happens if the school is not found in the map? Why do you think it is set to that value? If the school is not already present in the map, then it is

added to the map and initializes the 'count' variable to 1 for that school. It's set to 1 because it is being used to count the number of times the University shows up in the map.

6. What happens if it already found the particular university in the map? Then the else statement executes and the count is incremented by 1. (it is not added to the map again)
7. Navigate to the main.cpp file and observe the for loop under Part 5. Where is the function call to GetCountsByUniversity()? It is inside the for loop - data.GetCountsByUniversity() is called to get a map of counts of students for each university
8. What type of data in the map do you suppose i.first will print? How about i.second? I.first will print the key of each pair and i.second will print the associated value for each pair
9. Run the program to see if your theory was correct.
10. Navigate to the StudentData.cpp. Complete the GetCountsByState() function to get a count of how many students in are from each of the states.
11. Navigate back to the main.cpp function. Complete the loop to display all the counts for the states.
12. Here is the output from the first few states to verify that your code is working correctly…

```
States:
Alabama                 1996
Alaska                  2013
Arizona                 1938
Arkansas                1979
California              2022
Colorado                1967
Connecticut             1961
Delaware                1965
Florida                 2016
```

Part 6: Lambda's Practice

Observe and utilize maps by answering the following questions.

1. Navigate to the main() method in main.cpp. Comment out the function call to Part5() and remove the comment to the function call to Part6().
2. Navigate to the function Part6(). Observe the function call. What values are being sent to the filter function? 5000, Female, and Utah
3. Navigate to the StudentData.cpp file and observe the Filter function. What container is being used to store the results? A vector is being used
4. Observe the copy_if function. What vector is it using to copy the data from beginning to end? It's using the studentData_ vector
5. What command is used to insert the results into the vector "results"? back_inserter(results)
6. Observe the lambda function. Which variables are used in the capture clause? Id_range and gender
7. What is the parameter of the lambda expression? StudentInfo &s
8. Observe the id_range. For what range of values are getting returned? Id's greater than 'id_range' and less than the 'id_range' + 1000; (so this includes the id_range itself and up to, but not including id_range + 1000)
9. Run the program. View the output. Are all of the values returned within that range of Id values? Are all of the results the correct gender? Are all the results from the correct state? Yes, they are all within the range of Id values. Yes, they are all female. No they are not from the correct State.
10. Modify the lambda to return only the results of the given state.
11. Run the program to ensure you are now getting only results from Utah.

12. Observe the results. For what value are they sorted?They are sorted by the id_range value.
13. Observe the sort function. Are there any variables in the Capture clause? Why or why not? It doesn't need to capture any external variables because all it's doing is sorting based on the Id's, so it takes two parameters, s1 and s2, and returns true if s1 is less than s2 and false otherwise.
14. Observe the lambda Parameters. Why do you suppose there are two instances of the studentInfo class? Because these are the 2 parameters being compared in order to sort the elements, s1 and s2 are references to two StudentInfo objects in the vector. Their id numbers are being compared and sorted.
15. Modify the sort lambda so it is sorted by University.
16. Run the program. Your output should look similar to the following.