CSC 323 Algorithm Design and Analysis
Spring 2016
Instructor: Dr. Natarajan Meghanathan
Project 5
Determining the Common Elements across All Arrays using Hash Tables and Java Collection Classes

Due: March 31, 2016, 1 PM **Maximum Points: 100**

The **objective** of this project is to find the common elements among a given number of arrays. You should use the Java Collections classes (like Vector and TreeMap) to implement this project.

You will **input values** for the following parameters from the user:
* Number of arrays, N
* Number of elements per array, E
* Hash Table size, P
* Maximum value of any element in the array, M

Design Idea
You will generate the contents of the array using a random number generator (Random class in the java.util package). The values of the integers in each array should be bounded above by the maximum value of the elements (input by the user). An integer should not appear more than once in an array; but it may occur in more than one array (remember: the objective of the project is to find integers that appear in all the arrays). I suggest using separate random number generators for each array and a different seed for each of these random number generators.

After populating all the arrays with the randomly generated integers (as described above), create a Vector of size equal to the Hash Table Size, P. The Vector will serve as our hash table. Initialize each entry of this Vector with an empty TreeMap object. Now, iterate through each of the arrays. Compute the hash value for every integer *x* in each of the arrays. Use the following hash function: $h(x) = x \bmod P$, where P is the Hash Table Size. Now, index the Hash Table using the hash value $h(x)$ and store the integer *x* in the Hash Table as a <Key, Value> pair in the TreeMap at index $h(x)$: the Key is the integer *x* and Value is the number of times you have encountered (frequency) the integer *x* so far across all the arrays. As you encounter the integer *x* in the different arrays, you increment the frequency value by 1.

After filling the Hash Table with the <Integer, Frequency of Appearance> pairs for every integer (indexed into the appropriate TreeMaps) across all the arrays, index into each of these TreeMaps and identify integers whose frequency of appearance equals the number of arrays (N) in the program. Such integers are the common integers that appear across all the arrays. Collect all of these common integers into a separate vector and print the contents of this vector. Note: Sometimes, you may not be able to find any common elements. In that case, run your program few more times to see if you can get an output that displays one or more common elements.

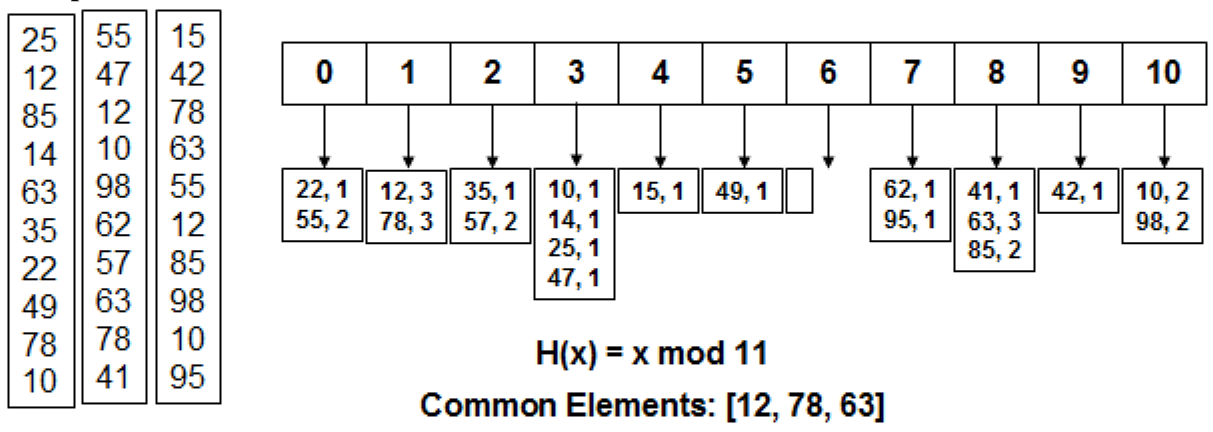
Time Evaluation: For each of the following combinations of the input values, you should also determine the total time it takes for your program. Use the System.nanoTime( ) method of the System class that returns the number of nano seconds since Jan 1, 1970, as a value of type long. Call this method after getting the inputs from the user and before beginning any other processing. Likewise, after completing all the processing and before printing the vector of common elements, call the System.nanoTime( ) method. The difference between the two time instants is the run-time of your program in nano seconds. You need to then convert the time difference to milliseconds or microseconds and report the values.

Input Values

Student Name	# Arrays	# Elements per Array	Hash Table Size	Maximum Value of any Element in the Array
Leon Anderson	4	200	53	500
Michael Brown	4	200	101	500
Aniekan Essien	6	200	59	500
Raven Lawrence	6	200	107	500
Shawndon Portis	8	200	67	500
Allaysia Roberts	8	200	103	500

Outputs: The common elements of the arrays and the run-time.

Example:



Video: Record your explanation of the complete program and show the execution for the inputs specified above. You should explain how the random integers are generated for all the arrays, how the hash table is populated with the <integer, frequency of appearance> pairs for every integer across all the arrays, and how you go through the hash table to list the common elements across all the arrays. Your explanation should last for at least 7 minutes. You could try using one of the **desktop recording software** (or anything of your choice): CamStudio: http://sourceforge.net/projects/camstudio/files/legacy/
Debut: http://www.nchsoftware.com/capture/index.html

Sample Programs and Video Links: Refer to the sample programs posted in the course website along with this project description on Vector and TreeMap.
Video on Vector Example: http://www.youtube.com/watch?v=aiZHl18TXqE
Video on TreeMap Example: http://www.youtube.com/watch?v=sGrHlTsTmxY

What to submit: (1) **A Desktop recorded video** (as explained above). (2) A **softcopy** of your complete code and a screenshot of the result obtained for the input specified above, emailed to me.

Note that even though I am not specifying a minimum time for your video, your video explanation is expected to last at least for 8-10 minutes and should cover all of the above required explanations.

Note that the contents of the desktop/programs captured through your video should be clearly readable.

Submission of the Video: Upload the video to your Google Drive and share it to natarajan.meghanathan@jsums.edu.