Re-submit Assignment

DVC Schedule v.3, Lab Assignment 9

Due Nov 1 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** cpp and h

Part 1

Write and fully test an AssociativeArray template, with all the public interface functions presented in this module. Name its file as **AssociativeArray.h**. Test but do *not* submit your test driver CPP. At this point in the semester you should know how to fully test a template, so *do* it, but don't *submit* it. Submit the H along with part 2's CPP.

Part 2

Write a new version of the DVC schedule program that (1) makes use of your AssociativeArray template and (2) expands the results to list the number of *courses* offered per subject code (not the number of *sections*) and the number of sections of *each course*. Like this:

```
ADJUS, 16 course(s)
  ADJUS-120, 191 section(s)
  ADJUS-121, 57 section(s)
  ADJUS-122, 40 section(s)
  ADJUS-130, 24 section(s)
  ADJUS-203, 20 section(s)
  ...
...
SPTUT, 1 course(s)
  SPTUT-020NC, 12 section(s)
TAGLG, 2 course(s)
  TAGLG-155, 5 section(s)
  TAGLG-156, 3 section(s)
```

Base the duplicate checking on the (fast-running) DVC Schedule v.2, and not the (slow-running) DVC Schedule 1. This should run as fast as DVC Schedule v.2.

Use any combination of your own StaticArray, DynamicArray, and AssociativeArray templates, and *no STL containers* for duplicate checking or subject code and course counting, but be sure to use your AssociativeArray for at least one of your data structures in the solution. If you do use your StaticArray or DynamicArray, include their H files in your file submissions.

Do *not* use struct-based objects like you did in the previous solutions. Using your AssociativeArray, structs should no longer be necessary. Instead, for counting use an AssociativeArray with course as the "key" and #of sections of that course as "value". Use that as the "value" *inside* another AssociativeArray with subject code as the "key". For duplicate checking, you can track term-sections with an AssociativeArray of string-AssociativeArray as explained in the module. Or use some other way of your own design, using your own StaticArray, DynamicArray, and AssociativeArray templates.

Work this all out on paper before coding anything, making sure to decide upon names for the AssociativeArrays, and the code for retrieving subject codes, courses, and counts.

Submit **DvcSchedule9.cpp**, **AssociativeArray.h**, and if you use them, resubmit your previous **StaticArray.h** and/or **DynamicArray.h**.

HINT: Is the output so lengthy that you cannot scroll to the top of it? Use the **more** option on the command line. For a Windows PC running a program named **a.exe** it's:

```
a.exe | more
```

Lab Assignment Rubric

Criteria	Ratings							Pts
Fully accurate results, following all specifications **view longer description**	Works the first time. 70.0 pts	Works on the 2nd try 65.0 pts	Works on the 3rd try 60.0 pts	Works after 4 or more tries. 50.0 pts	Doesn't work after 2 weeks. Partial credit. 20.0 pts	Not submitted within two weeks of the due date. 0.0 pts	Work is not original -- appears to be a marked-up copy of the work of another or previous student. 0.0 pts	70.0 pts
Submits all work on time, fully complete if not fully correct. **view longer description**	Submitted on time 20.0 pts	Submitted on time, but one or more files are missing or not correctly named. 16.0 pts	Submitted on time, but with missing identification in one or more submitted CPP or H files. 15.0 pts	Submitted on time but not fully complete. 10.0 pts	Late or wholly incomplete! 0.0 pts			20.0 pts
Well-organized and professional quality code. **view longer description**	Fully meets expectations 10.0 pts	Mostly meets expectations, just needs to be a bit more careful. 8.0 pts	Many areas are well done, but there are a lot of areas that need work. 6.0 pts	Getting there, but needs to be a lot better. 3.0 pts	Needs a lot of work. See the instructor for guidance. 0.0 pts			10.0 pts
								Total Points: 100.0