HashTable Template, Lab Assignment 10

Re-submit Assignment

Due Nov 8 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** cpp and h

This is a 3-part assignment, involving 2 programs -- a new DvcSchedule, and **Conway's Game Of Life (https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life)** .

PART 1, A HashTable Template

Write and fully test a HashTable template, with the *same public interface* functions as the AssociativeArray (except that if you wrote a capacity *setter*, do *not* include that). Name its file as **HashTable.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.

As a first application of your HashTable, use it in the classic ***Game Of Life Simulation***. Right-click here: **GameOfLife.cpp**, to download the already-completed main program that should work with your **HashTable.h**.

Here's an interesting input sequence to use:

```
1 1 2 2 3 3 3 4 3 5 -1 -1
```

The above should reach an unchanging "steady state" like this after a few generations:

```
                         X
                        X X
                        X  X
                         XX
```

Here's another:

```
1 1 2 2 3 3 4 4 3 4 4 3 5 2 1 4 -1 -1
```

The 24th generation should look like this:

```
                     XXX     XXX
                   XXXXX     XXXXX
                   XX    X X    XX
                  XXXX  XXXXX  XXXX
                        X X X
                 X X    X X X    X X
                  XXX X       X XXX
                      XXX X XXX
                       X     X
                         X X

                         XXX
```

Note that the 24th generation is symmetric. Because of the rules for the Game Of Life, one should expect that ALL generations beyond the 24th will also be symmetric. Make sure that you test this *well beyond* the 24th generation to convince yourself that your hash table class works right.

Submit **HashTable.h** only.

Part 2, DVC Schedule v.4

Write **DvcSchedule10.cpp** by modifying *DvcSchedule4.cpp (DVC Schedule v.1)*, applying your HashTable template for duplicate checking and counting. *This version should run lightning fast!* It may also require a larger "stack size" than the default 1MB -- during grading, the professor will use a large enough stack for whatever your program may need.

You may use STL containers for sorting before outputting your results. Do ***NOT*** use your **StaticArray.h** and/or **DynamicArray.h**.

But watch out! If you create hashing functions that do not span the capacity of your HastTables, then much of that capacity could be wasted space, and you won't see O(1) behavior -- meaning this will *not* run lightning fast. For example, if your key is 10-character strings, and you simply sum their ASCII values, the maximum hash code will be about 1000. Used in a 100,000-capacity hash table, it uses only 1% of that capacity!

Submit **DvcSchedule10.cpp**. You should already have submitted HashTable.h in part 1 above.

Part 3, The STL map Template

Use the STL ***map*** template to do the Game Of Life simulation, replacing your HashTable.h.

Rewrite the downloaded GameOfLife.cpp from part 1, naming the new program **GameOfLifeMap.cpp**. There should be no include for *your* HashTable in this version -- just the STL map. Here are the differences:

1. The STL map does not use a **hashCode** function, so you won't need that.
2. The STL map does not need **operator==** for its key, but it *does* need **operator<** instead.
3. The STL map does not have **containsKey**, so you'll have to find its equivalent in the STL documentation online.

Submit **GameOfLifeMap.cpp** only.

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