CS 3114 Summer 2015 GIS **Student:** _____

GTA: _____

Submission date: (from the submit log, not a file timestamp)	Sub #:	Deduction for: fundamentals	0 /250
		design/engr	0 /90
		documentation	0 /10
		correctness	0 /200
		Total score out of 300	0 /300

Fundamental Requirements Have the student show you the relevant areas in his/her implementation		**Item Deduction**
Required data structures elements[1]:		
PR quadtree – does not use a PR quadtree	-100	
Hash table – does not use a hash table	-100	
Buffer pool – does not use a buffer pool	-50	

Design and Engineering (-90 points maximum) Have the student show you the relevant areas in his/her implementation		**Item Deduction**
PR Quadtree implementation[2]:		
quadtree internal nodes store coordinate/boundary data for coordinate regions	-20	
quadtree internal nodes store anything <u>else</u> besides four node pointers	-10	
quadtree leaf nodes store node pointers	-20	
quadtree leaf nodes do not store specified "bucket" of index objects	-10/-30	
not easy to modify bucket size	-15	
region search not properly optimized	-20	
region search uses Java library `Rectangle.intersects()` or similar	-30	
Hash table implementation[3]		
does not use an array or `Vector` generic for the physical storage of the table	-30	
does not use quadratic probing with (k^2+k)/2 to resolve collisions	-20	
table sizes do not conform to specification	-20	
Buffer pool implementation[4]		
does not employ LRU replacement policy	-20	
does not use 10 slots to cache records	-10	
Feature name/state and location indices[5]:		
no wrapper class for the quadtree/hash table that holds the index entries	-20	
index data (e.g., feature name/state and set of file offsets) not encapsulated within a class (It's OK if public data is used here.)	-10	
Index stores complete GIS records	-25	
General infrastructure[6]		
no overall controller class or command processor class	-10	
no class to encapsulate logic of retrieving next command from script file	-10	
no class to represent a GIS record (`String` is not acceptable)	-10	

Comments:	**Total deduction for this section:**

Internal Documentation (-10 points maximum)	
Spot-check comments in a randomly selected source file: Inadequate function/method headers: -5 (statement of purpose, parameter comments, pre/post conditions) Inadequate internal comments in function bodies: -5	
Comments:	**Total deduction for this section:**

Correctness of Program Operation (-190 points maximum)	
Basic test of index-building (-30 points maximum) Command line: `db01.txt DemoScript01.txt Log01.txt` General log issues: Commands not echoed and numbered in log file -10 Log format is badly organized, hard to read -10 PR quadtree integrity shown in display of the location index[7]: Nodes are not organized as a PR quadtree, or not clear from display -30 Doesn't have 13 leaf nodes -10 Doesn't show a record holding two offsets (like 1366 and 3008 in mine at line 49) -10 Hash table integrity shown in display of the feature ID index[8]: Index entries are not organized as a hash table, or not clear from display -30 Doesn't have 25 full slots -10 Did not create a db file containing the imported records[9] -30	
Comments:	**Total deduction for this section:**

Test of simple searching with a small dB file (-20 points maximum)
Command line: `db02.txt DemoScript02.txt Log02.txt`
 	
Basic searches[10]: -3 per search if anything is wrong
 	
`what_is United States Mountain CO`
`what_is Cottonwood Creek CO`
 	
`what_is_at 380145N 1074019W`
`what_is_at 375437N 1074146W`
 	
`what_is_in -c 380122N 1074015W 15 15`
`what_is_in  380122N 1074015W 15 15`
`what_is_in -l 380122N 1074015W 15 15`	

Comments:	**Total deduction for this section:**

Test of multiple imports[11] (-20 points maximum) Command line: `db03.txt DemoScript03.txt Log03.txt` Runtime crash during the test -10 Searches: `what_is_in 381257N 0794039W 120 60` -10 `what_is_in 381621N 0794457W 1200 30` -10	
	Total deduction for this section:
Test of the buffer pool with a small dB file[12] (-30 points maximum) Command line: `db04.txt DemoScript04.txt Log04.txt` Runtime crash during the test -10 Buffer pool contents and ordering: -4 per display if anything is wrong `Command:` `3` -2 `8` -4 `14` -10 `16` -4 `18` -4 `20` -4 `22` -6 `25` -6 `30` -6 `35` -6	
Comments:	**Total deduction for this section:**
Test for search failures[13] (-20 points maximum) Command line: `db05.txt DemoScript05.txt Log05.txt` Runtime crash during the test -10 Report incorrect records (<u>none</u> should match) -5 per search Don't log an informative message but don't log incorrect records either -5	
Comments:	**Total deduction for this section:**

More varied test of region search[14] (-30 points maximum)

Command line: `db06.txt DemoScript06.txt Log06.txt`

Runtime crash during test -10

`Command` -4 each if anything is wrong

2
3
4
5
6
7
8
9

Comments:

Total deduction for this section:

Test with large database file[15] (-50 points maximum)

Command line: `db07.txt DemoScript07.txt Log07.txt`

Runtime crash during the test -10

Any sort of failure in any of the searches

```
what_is  Nester Draw               NM                    -4 each
what_is  Screaming Left Hand Turn  NM
what_is  Window Rock               NM
what_is  Buena Vista               NM
;
; Now do some location searches:
what_is_at  363957N  1054049W                            -4 each
what_is_at  351018N  1034328W
what_is_at  362846N  1085222W
what_is_at  334236N  1055604W
;
; And some region searches:                              -6 each
what_is_in  362846N  1085220W  120 120
what_is_in  333859N  1062731W  120 120
what_is_in  345326N  1073457W   60  60
```

Comments:

Total deduction for this section:

Other Adjustments

Other Issues

Note any problems you noticed that weren't covered above and suggest a penalty for them:

Notes:

1 This is simply checking whether the solution actually implements the three mandatory data structures. You are concerned yet with whether they are implemented correctly. Be careful of situations where a very incomplete implementation is supplied, but not actually used. There should be enough of an implementation to convince you that the student has actually made a serious attempt to complete the requirements.

If the deductions for the quadtree or hash table apply, either the student will have substituted some other structure, probably something much simpler, or else the student will not have a working solution.

If the student is penalized here, try to avoid double-jeopardy in the later sections. For example, if the student did not implement a buffer pool, skip the test of the buffer pool when you test the functionality (and do not penalize the student for that test, but do enter a comment for that test indicating it was skipped, and why).

2 The discussion in class made it clear that internal nodes store only pointers to other nodes, and that leaf nodes do not store pointers, and that it is not acceptable to store the boundaries of the region a node represents in that node.

The specification and class discussions were perfectly clear that a bucket PR quadtree is to be used with a bucket size of 4. Deduct 30 points for having no bucket at all; deduct 10 points for having a structure to store multiple records in the leaf node, but not providing the specified bucket size.

Look at the region search code and determine whether the student is correctly comparing the boundaries of the search region to the boundaries of each subtree root node in order to decide whether to actually search the subtree.

For "ease of changing the bucket size", look at the code. It should be possible to change the bucket size by altering a single line of code (a constructor call, the declaration of a static final class member, etc.).

3 They cannot possibly achieve Theta(1) search cost unless they use an array or an array-based Java generic, like Vector or ArrayList. The required probe function and table sizes were given in the specification.

4 It may be easier to verify the first item by waiting to check the output from the test of the buffer pool. They may use any underlying physical structure to organize the slots.

5 The point of the wrapper classes is to provide an appropriate interface for query transactions. This was discussed in class, and shown in the posted solutions for the design homework. Note that it is absolutely not an acceptable design to use the naked quadtree or hash table in place of this requirement.

The issue in the second item is whether they've properly encapsulated the data for the index entry. For the location index, it is acceptable if they separate the location data from the set of file offsets, as long as they use a single object for each (so two objects altogether).

6 These were all shown in the posted solutions to the design homework. For the first item, it's OK if they embed the handling of each command inside a specialized class for that command.

7 They must display the PR quadtree in such a way that you can verify it looks like a proper PR quadtree. It must be possible to tell which nodes are in each level, and how what the parent/child relationships are. It's OK if they don't use the same format I used in my logs, but it MUST be clear to you that the structure is really a PR quadtree. I would expect their PR quadtree to have the same number of nodes and the same number of levels as mine, but they may display the quadrants in a different order. Record offsets might be slightly different than mine; if so that's OK.

8 The hash table display should at minimum show the filled slots, including both the file offset for the record and some indication of what the record is. The simplest approach is to just show the entire line from the file, but it's acceptable to show less than that as long as it's clear what the record is.

It is entirely possible that they will show a few records in different slots than I do; that's OK. But most of them should be in the same slots since there won't be many collisions with so small a test data set.

9 It's possible that the student's db file will be organized differently than mine. That's OK, as long as it contains exactly 61 records.

10 For each search, they should report exactly the same record(s) that I do. It's OK if they don't show every record field, but they must show enough for you to check the results. I don't think there is much reason for partial credit on an individual search unless you want to deduct points for not showing enough record fields.

11 Each of the region searches is designed so that the results should include records from both of the files that were imported, so if any records are missing, that probably indicates the student did not handle the second import correctly. Give no partial credit for either search.

12 The student must display the records in MRU to LRU order. The searches were chosen so that each should yield a single match. So, the contents of the student's pool should be the same as mine, and they should be in the same order. I would not give any partial credit for each search.

If the search results are incorrect, it's probably not going to be possible to verify that the student implemented the buffer pool correctly; do not give credit for anything here unless you are convinced by the output that the implementation of the buffer pool is correct.

13 This is all about how, or if, they handle searches that do not find any matching records. There is a high probability that they will have runtime errors on this test, if they have them anywhere. For each search, there are three possible outcomes:

- They log a message indicating nothing was found, and do not have a runtime error.
- They log incorrect records that do not match the search criteria, and do not have a runtime error.
- They have a runtime error.

If they have any runtime errors, deduct points as specified and then evaluate their output, if there is any.

If they report records that really do not match the search criteria, they should lose 5 points for each such search. Finally, if they don't report any incorrect records, and don't have a runtime error, but don't log an informative message, they should lose 5 points (not per test).

14 With region searches, the order in which they report the matching records does not matter. If a search results in more than 5 matches, just verify that they report the correct number of records, unless they're getting wrong results for the searches with a small number of matches.

15 There are 11 separate searches here, and only 50 points to go around. Assign 4 points each to the `what_is` and `what_is_at` searches, and don't give any partial credit (if you see a case where you think that's too harsh, ask me about it). Assign 6 points to each of the `what_is_in` searches, and evaluate them as in the previous test.