CS 3114 Fall 2010 GIS **Student:** _____

GTA: _____

Submission date: (from the submit log, not a file timestamp)	Sub #:	Deduction for: fundamentals	xx /250
		design/engr	xx /100
		documentation	xx /10
		correctness	xx /190
		Total score out of 300	xxx /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 (-100 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	
Hash table implementation[3]		
does not use an array for the physical storage of the table	-30	
does not use some form of quadratic hashing to resolve collisions	-20	
Buffer pool implementation[4]		
does not employ LRU replacement policy	-20	
does not use 20 slots to cache records	-10	
Feature name/state and location indices[5]:		
no wrapper class for the container 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:**

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. They may any form of quadratic probing they like, and they may revert to a backup strategy, like linear probing, after a reasonable number of unsuccessful steps using quadratic probing.

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.