University of California, Berkeley
College of Engineering
Computer Science Division – EECS

Fall 2010 Prof. Michael J. Franklin

MIDTERM I

CS 186 Introduction to Database Systems

NAME:_____ **STUDENT ID:**_____

IMPORTANT: Circle the last two letters of your class account:

cs186 a b c d e f g h i j k l m n o p q r s t u v w x y z

a b c d e f g h i j k l m n o p q r s t u v w x y z

DISCUSSION SECTION DAY & TIME:_____ **TA NAME:** _____

This is a **closed book** examination – but you are allowed one 8.5" x 11" sheet of notes (double sided). You should answer as many questions as possible. Partial credit will be given where appropriate. There are 100 points in all. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time-consuming than others.

Write all of your answers directly on this paper. **Be sure to clearly indicate your final answer** for each question. Also, be sure to state any assumptions that you are making in your answers.

GOOD LUCK!!!

Problem	Possible	Score
1. Buffer Manager	**30**	
2. B-Trees	**24**	
3. Hash Indexes	**24**	
4. Formal Query Languages	**22**	
TOTAL	**100**	

SID:_____

Question 1 – Buffer Management [4 parts, 30 points total]

a) [15 points] Consider a database system with 4 buffer frames (A, B, C, D) and a file of 6 disk pages (1, 2, 3, 4, 5, 6). Assume that you start with an empty buffer pool. A sequence of requests is made to the buffer manager as described in the Request column (below). At certain times a Pin request is immediately followed by an Unpin request (represented as Pin/Unpin), but other times Pin and Unpin requests happen in an interlaced manner with other requests.

Fill in the following table showing the buffer contents after the completion of each operation using the **CLOCK** page replacement policy as described in the book and HW 1. For each page, indicate the pin count (PC) and for unpinned pages, indicate the value of the reference bit (true or false). You should mark unchanged buffer frames with "ditto" (“). As in HW 1 we make the following assumptions in the clock policy:

The pointer doesn't move when filling a free frame in buffer or on a buffer hit.

The pointer advances after replacing a buffer frame

Note: Total points for this problem is 15, (3 each for correct buffer contents at steps T4-T8)

Time	**Request**	**Buffer Frames**			
		A	B	C	D
T1	Pin 5	**5** PC=1	Empty	Empty	Empty
T2	Pin/Unpin 2	”	**2** PC=0 Ref = t	”	”
T3	Pin/Unpin 3	”	”	**3** PC = 0 Ref =t	”
T4	Pin 4				
T5	Pin 1				
T6	Unpin 5				
T7	Pin/Unpin 2				
T8	Pin 6				

SID:_____

Question 1 – Buffer Management (continued)

b) [5 points] Consider an access pattern where there are lookups of many different keys in a B+tree. Assume that the B+Tree is 4 levels deep and the buffer pool has only 3 buffer frames. Which replacement policy will provide the best hit rate and why? **Circle your answer and be sure to state why. List any assumptions you are making.**

a) LRU **b)** MRU **c)** Random **d)** All are equal

Why?

c) [5 points] Consider an access pattern where there are lookups of many different keys in a static hash file (with no directory). Assume that hash file is much bigger than the buffer pool and that the hash function works well on the given data. Which replacement policy will provide the best hit rate and why? **Circle your answer and be sure to state why. List any assumptions you are making.**

a) LRU **b)** MRU **c)** Random **d)** All are equal

Why?

d) [5 points] Consider a situation where the access pattern changes suddenly. For example, one set of pages is heavily accessed during the week, but starting at 11:59pm on Friday, a completely different set of pages becomes heavily accessed. Which of the following would you expect to react more quickly to the change in workload (i.e., have a better hit rate once the workload changes). **Circle your answer and be sure to state why. List any assumptions you are making.**

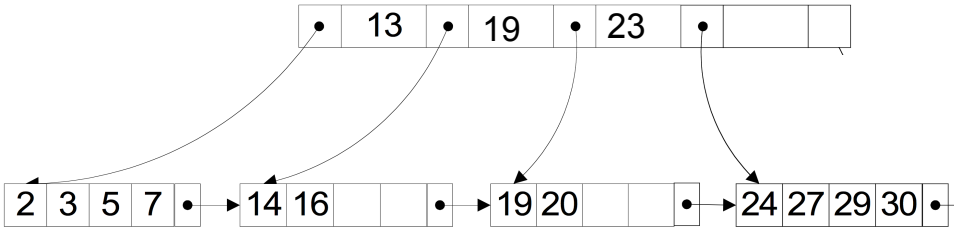
a) LRU **b)** LFU **c)** Not enough info to tell

Why?

SID:_____

Question 2 – B+Trees [6 parts, 24 points total]

Consider the following B+Tree with order d=2. Recall that the insert algorithm splits full leaf nodes it is inserting to rather than trying to avoid splits by redistributing data entries to neighbors.



a) [4 points] Which of the following key values for data entries, <u>if inserted on its own</u>, would cause a LEAF PAGE SPLIT (Circle all that apply):

12 13 18 23

b) [2 points] For the original tree above, what is the **minimum number of keys** that you could insert that would cause the ROOT NODE to split? **Give an example of a sequence of such inserts.**

c) [2 points] Considering the original B+tree above, which of the following is true?

(Circle one only, and briefly explain your answer)

A. No deletions have ever been performed on it.

B. One or more deletions have been performed on it.

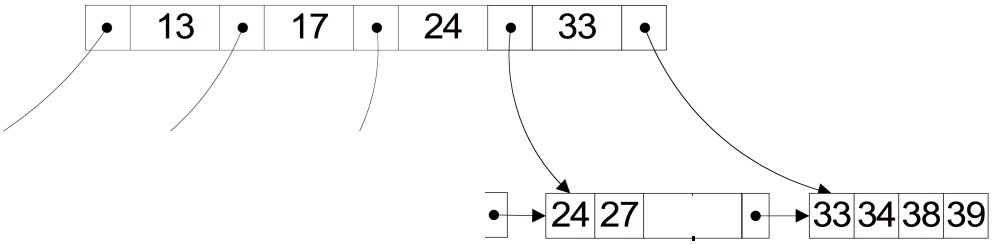
C. Not enough information to tell.

Why?

d) [4 points] STARTING WITH THE ORIGINAL TREE ABOVE, draw the B+Tree that results from deleting the data entry with the key 14. (you only have to draw the parts of the tree that change).

SID:_____

e) [6 points] Consider the following B+Tree with order d=2 (with some leaf nodes elided)



Draw the B+Tree that results from inserting a data entry with the key 40. (hint: recall that the root node is not required to have d keys in it).

f) [6 points (3 points each)] In the B+Tree algorithm covered in class, we treat the split of a leaf node differently than the split of an index node. Specifically, when splitting a leaf node we make a copy of the new separator key and insert that into the parent node, while when splitting a non-leaf node, we promote a key (i.e., move it) into the parent node. Your project partner argues that it would be simpler to treat all nodes the same – either by copying for all nodes, or promoting for all nodes. For each of these two options, would it work *without changing the basic B+Tree search algorithm*? If not, why not? If so, is there any disadvantage to doing it compared to the way we did it in class?

Option 1 –Copy for all Nodes: Does it work? YES or NO

If No, say why. If Yes list one disadvantage:

Option 2 –Promote for all Nodes: Does it work? YES or NO

If No, say why. Yes list one disadvantage:

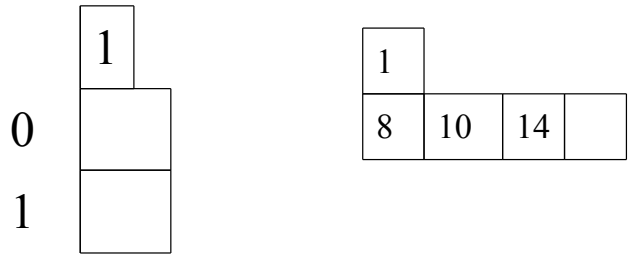
SID:_____

Question 3 – Hashing [5 parts, 24 points]:

a) [10 points] Extendible Hashing Consider the following 4 update operations.

operation no.	operation	key value (binary)
1	insert	25 (011001)
2	insert	46 (101110)
3	insert	18 (010010)
4	insert	22 (010110)

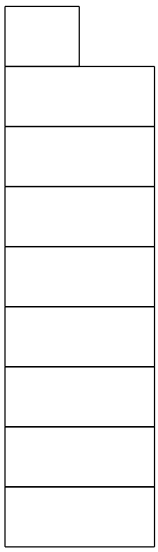
Now, consider an extendible hash structure with a hash function h(n) = n mod 2^{depth} and hold up to 4 entries, shown below. Draw the extendible hash structure and its contents after the 4 operations have occurred in the order shown. We recommend that you do your scratch work on this page at first. But, this page will not be graded. You MUST put your final answer on the following page!!



SID:_____

Final answer for Question 3(a) - Extendible Hashing:

- Only this page will be graded for question 3(a).
- The final structure should have a directory of size 8 so use the template below.
- Show all buckets and pointers
- Label the directory entries with their corresponding hash value (as on the previous page).
- Make sure to include local depths for all buckets and the global depth of the directory.

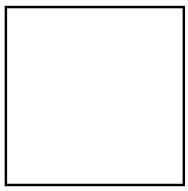


SID:_____

Question 3 – Hashing (continued)

Now, consider a *linear hashing* index with the following initial state:

Consider the sequence of inserts shown in the answer parts b and c.



operation no.	operation	key value (binary)
1	insert	20 (010100)
2	insert	46 (101110)
3	insert	18 (010010)
4	insert	23 (010111)

b) [4 points] Assuming the operations are performed in the order specified in the table, which of the four operations will cause the bucket pointed to by the "next" pointer (i.e., "bucket 1") to be split?

c) [3 points] What will be the value of "Next" after the split (from part b) happens (i.e., which bucket will be scheduled to be split on the subsequent overflow)?

d) [3 points] For the above structure, how many buckets will there be (ignoring any extra pages used for overflow) at the beginning of round 4 (i.e., when "Level" = 4)?

e) [4 points] In class we didn't specify if the data entries inside buckets (including overflow chains) should be kept sorted by key value or simply placed in the order they arrive. List one advantage and one disadvantage of keeping the bucket contents sorted. State which operations are affected.

Advantage (Pro):

Disadvantage (Con):

SID:_____

Question 4 – Formal Query Languages [3 parts, 22 points total]

Consider the following schema that records information about a university (Primary keys are underlined):

Student (sid, sname, address, age)

Course (cid, cname, credits)

Professor(pid, pname, address, age, department)

Grade(sid, cid, pid, grade)

Write queries to answer the following questions. Note that we don't care about efficiency here, but **points will be taken off for using unnecessary relations in your queries**.

a) **[8 points (4 each)]** Find the sid and sname of students who have received at least one A in a *four-credit* course.

i) Relational Algebra

ii) Relational Calculus

SID:_____

Recall the schema from the previous page (Primary keys are underlined):

Student (sid, sname, address, age)

Course (cid, cname, credits)

Professor(pid, pname, address, age, department)

Grade(sid, cid, pid, grade)

b) [8 points (4 each)] Find the sids of students who have taken at least one course and have received a grade of "A" in every course they have taken.

i) Relational Algebra

ii) Relational Calculus

c) [6 points] Find the pids of professors who have taught **exactly two** different classes. Use your choice of Relational Algebra or Relational Calculus.