Database Concepts- Assignment 8
Due: December 5 by 11:59pm

1. (a) Consider the following parameters:

block size	=	4096 bytes
block-address size	=	8 bytes
block access time	=	20 ms (micro seconds)
record size	=	100 bytes
record key size	=	9 bytes

Assume that there is a B^+-tree, adhering to these parameters, that indexes 10 million records on their primary key values.

i. Specify (in ms) the minimum time to determine whether a record with key k is in the B^+-tree.

ii. Specify (in ms) the maximum time to determine whether a record with key k is in the B^+-tree.

iii. How many records would there need to be to increase the minimum time to determine whether a record with key k is in the B^+-tree by approximately 50 ms?

Show all the intermediate computations leading to your answer.

(b) Consider the following B^+-tree of order 2 that holds records with keys 1, 6, 8, and 10. (Observe that (a) an internal node of a B^+-tree of order 2 can have either 1 or 2 keys values, and 2 or 3 sub-trees, and (b) a leaf node can have either 1 or 2 key values.)

```
          -----------
          | /  8  \ |
          -----------
           /         \
          /           \
     ----------     ------------
     | 1   6  |--->| 8      10 |
     ----------     ------------
```

i. Show the contents of your B^+-tree after inserting records with keys 5, 9, 12, and 3, in that order.

ii. Starting from your answer in question 1(b)i, show the contents of your B^+-tree after deleting records with keys 1, 12, 3, and 9, in that order.

2. (a) Consider an extensible hashing data structure wherein (1) the initial global depth is set at 1 and (2) all directory pointers point to the same **empty** block which has local depth 0. So the hashing structure looks like this:

```
global-depth 1       local-depth 0
    ------         -----------------
  0 |  --|-------->|               |
    ------         |               |
  1 |  --|-------->|               |
    ------         -----------------
```

Assume that a page can hold at most two records.

i. Show the state of the hash data structure after each of the following insert sequences:

A. records with keys 1 and 5.

B. records with keys 2 and 3.

C. records with keys 4 and 7.

ii. Starting from the answer you obtained for Question 2(a)i, show the state of the hash data structure after each of the following delete sequences:

A. records with keys 1 and 3.

B. records with keys 2 and 4.

C. records with keys 5 and 7.

iii. Give an example where the insertion of a record in an extendible hash data structure can result in the recursive doubling of the directory of the hash data structure.

3. Consider a database wherein we maintain 2 data files:

   (a) a file STUDENT which keeps student records of the form (SId, Sname, major; SId is a key),

   (b) an ENROLLMENT file which keeps records of the form (SId, Cno).

   Specify a file organization for each of these files which would enable efficient processing of the following query

   "Find the course numbers of courses taken by students who major in a field with a name that begins with the letter 'A'." (For example African-Studies, Anthropology, Archeology etc.)

   Argue why your file organization provides efficient processing of this query.

4. Let $R(A, B)$ and $S(B, C)$ be two relations and consider their natural join $R \bowtie S$.

   Assume that R has 200000 records and that S has 10000 records.

   Furthermore, assume that 20 records of R can fit in a block and that 10 records of S can fit in a block.

   Assume that you have a main-memory buffer with 51 blocks. (One of these block is reserved for output purposes.) In the following questions, you should consider making maximum use of the buffer.

   (a) How many block IO's are necessary to perform $R \bowtie S$ using the nested-loops join algorithm? Show your analysis.

   (b) How many block IO's are necessary to perform $R \bowtie S$ using the merge-join algorithm? Show your analysis.

   (c) How many block IO's are necessary to perform $R \bowtie S$ using the hash-join algorithm? Show your analysis.

5. Give an example of two conflict-equivalent, but different, non-serializable schedules.

6. Give an example of a serializable schedule which is conflict-equivalent with two different serial schedules.

7. Consider the following transactions:

```
T1:  read(A);
     read(B);
     if A = 0 then B := B+1;
     write(B).

T2:  read(B);
     read(A);
     if B = 0 then A := A+1;
     write(A).
```

Let the consistency requirement be $\texttt{A} = 0 \vee \texttt{B} = 0$, and let $\texttt{A} = \texttt{B} = 0$ be the initial values.

(a) Show that each serial schedule involving transaction T1 and T2 preserves the consistency requirement of the database.

(b) Construct a schedule on T1 and T2 that produces a non-serializable schedule.

(c) Is there a non-serial schedule on T1 and T2 that produces a serializable schedule. If so, give an example.

(d) i. Add lock and unlock instructions to T1 and T2, so that they observe the two-phase locking protocol, but in such a way that interleaving between operations in T1 and T2 is still possible.

   ii. Can the execution of these transactions result in a deadlock? If so, give an example.