National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Advance Database Concepts**	**Course Code:**	**CS4064**
	Program:	**BS (Computer Science)**	**Semester:**	**Spring 2023**
	Out Date:	**27-Mar-2023**	**Total Marks:**	
	Due Date:	**Tue 4-Apr-2023 *(Start of class)***	**Weight:**	
	Section		**Page(s):**	**2**
	Assignment:	**3 (Indexing Structures)**		

Instructions:

- *This assignment is an individual assignment.*
- *You are required to submit the hard copy of your assignment at the start of your class.*
- *Use any valid assumption where needed.*
- *For any query, please contact your TA.*

Take the following assumptions for the block size and file size to solve the questions:

Q1. Block Size **B=4096 b**ytes and File Records **r=10m**illion
Q2. Block Size **B=8192 b**ytes and File Records **r=10b**illion

A block pointer is P = 6 bytes long and a record pointer is P_R = 7 bytes long. A file has above records of fixed length (un-spanned). Each record has the following fields: NAME (30 bytes), SSN (9 bytes), DEPARTMENTCODE (9 bytes), ADDRESS (40 bytes), PHONE (9 bytes), BIRTHDATE (8 bytes), SEX (1 byte), JOBCODE (4 bytes), SALARY (4 bytes, real number). An additional byte is used as a deletion marker.

a. Suppose that the file is *ordered* by the key field SSN and we want to construct a *primary* index on SSN. Calculate (i) the index blocking factor bfr_i (which is also the index fan-out *fa);* (ii) the number of first-level index entries and the number of first-level index blocks; (iii) the number of levels needed if we make it into a multilevel index; (iv) the total number of blocks required by the multilevel index; and (v) the number of block accesses needed to search for and retrieve a record from the file given its SSN value using the primary index.

b. Suppose that the file is not *ordered* by the key field SSN and we want to construct a *secondary* index on SSN. Repeat the previous (part a) for the secondary index and compare with the primary index.

c. Suppose that the file is not *ordered* by the non-key field DEPARTMENTCODE and we want to construct a *secondary* index on DEPARTMENTCODE, with an extra level of indirection that stores record pointers. Assume there are 20000 distinct values of DEPARTMENTCODE and that the EMPLOYEE records are evenly distributed among these values. Calculate (i) the index blocking factor bfr, (which is also the index fan-out *fa);* (ii) the number of blocks needed by the level of indirection that stores record pointers; (iii) the number of first level index entries and the number of first-level index blocks; (iv) the number of levels needed if we make it into a multilevel index; (v) the total number of blocks required by the multilevel index and the blocks used in the extra level of indirection; and (vi) the approximate number of block accesses needed to search for and retrieve all records in the file that have a specific DEPARTMENTCODE value, using the index.

d. Suppose that the file is *ordered* by the non-key field DEPARTMENTCODE and we want to construct a *clustering index* on DEPARTMENTCODE that uses block anchors (every new value of DEPARTMENTCODE starts at the beginning of a new block). Assume there are 20000 distinct values of DEPARTMENTCODE and that the EMPLOYEE records are evenly distributed among these values. Calculate (i) the index blocking factor bfr, (which is also the index fan-out *fa);* (ii) the number of first-level index entries and the number of first-level index blocks; (iii) the number of levels needed if we make it into a multilevel index; (iv) the total number of blocks required by the multilevel index; and (v) the number of block accesses needed to search for and retrieve all records in the file that have a specific DEPARTMENTCODE value, using the clustering index (assume that multiple blocks in a cluster are contiguous)

e. Suppose the file is not ordered by the key field Ssn and we want to construct a B$^+$-tree access structure (index) on SSN. Calculate (i) the orders p and p leaf of the B$^+$-tree; (ii) the number of leaf-level blocks needed if blocks are approximately 69% full (rounded up for convenience); (iii) the number of levels needed if internal nodes are also 69% full (rounded up for convenience); (iv) the total number of blocks required by the B$^+$-tree; and (v) the number of block accesses needed to search for and retrieve a record from the file--given its SSN value--using the B$^+$-tree.

f. Repeat (part e), but for a B-tree *rather than for a* B$^+$-tree. Compare your results for the B-tree and for the B$^+$-tree.

Q3. Consider a DBMS that has the following characteristics:

- 1KB fixed-size blocks
- 12-byte pointers
- 56-byte block headers

We want to build an index on a search key that is 8 bytes long. Calculate the maximum number of records we can index with a

a. 3 Level B$^+$- tree index (including the root level)
b. 3 Level B-tree index (including the root level)

Q4. Assume a relation *R (A, B, C)* is given; Suppose *A, B, C* are integer type values. Relation *R* is stored as an un-ordered file (un-spanned) on key field *A* and contains 5000 data blocks. Assume there is B$^+$- tree access structure (index) on *A* of height x=4 (root, 2 intermediate layer, leaf). Moreover, one node of the B$^+$-tree is stored in one block on the disk.

Estimate the number of block fetches needed to compute the following queries:

a. *SELECT *   FROM R WHERE A = 777;*
b. *SELECT C   FROM R WHERE A = 111 AND B = 3;*
c. *SELECT *   FROM R WHERE A = 111 OR A = 3;*
d. *SELECT *   FROM R WHERE A > 100;*
e. *SELECT COUNT(*)   FROM R WHERE A > 100;*