RollNo: ______________ **Name:** ________________________

National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Advanced Database Concepts**	**Course Code:**	**CS451**
Program:	**BS(Computer Science)**	**Semester:**	**Spring 2018**
Duration:	**60 Minutes**	**Total Marks:**	**30**
Paper Date:	**13-Apr-18**	**Weight**	**12.5%**
Section:	**CS**	**Page(s):**	**5**
Exam:	**Midterm-II**		

Instruction/Notes: Scratch sheet can be used for rough work however, all the questions and steps are to be shown on question paper. You may use backside of paper. ***No extra/rough sheets should be submitted with question paper***. You will not get any credit if you do not show proper working, reasoning and steps as asked in question statements.

Q1. *(8 points)* Assume a relation R (A, B, C) is given; R is stored as an ordered file (un-spanned) on non-key field C and contains 100,000 records. Attributes A, B and C need 10 bytes of storage each (i.e. record size= 30), and blocks have a size of 512 Bytes. Each A value occurs at an average 5 times in the database, each B value occurs 50 times in the database, and each C value occurs 5000 times in the database. Assume there is no index structure exists. Estimate the number of block fetches needed to compute the following queries (where C_a, C_b, C_{c1} and C_{c2} are integer constants):

a) SELECT B, C FROM R WHERE A = C_a;
b) SELECT COUNT(*) FROM R WHERE B = C_b;
c) SELECT A, B FROM R WHERE C = C_{c1};
d) SELECT A, B FROM R WHERE C = C_{c1} OR C = C_{c2};

Ans: bfr=512/30=17; b=100,000/17= 5883
a) O(b) = 5883
b) O(b) = 5883
c) O(log(b) + s/bfr - 1) = O(13 + 5000/17 – 1) = O(13 + 295 - 1) = 307
d) 2 * 307 (i.e. same cost as of part-c)= 614

RollNo: ______________ **Name:** ________________________

Q2. *(1+1+1+1+4= 8 points)* Consider a disk with block size *B=1024* bytes. A block pointer is *P=6* bytes long, and a record pointer is $P_R=7$ bytes long. A file has *r=1,000,000* EMPLOYEE records of fixed-length. Record length R is *115* bytes long and DEPTCODE field is *15* bytes long.

Suppose the file is ordered by the non-key field DEPTCODE and we want to construct a clustering index on DEPTCODE that uses block anchors (every new value of DEPTCODE starts at the beginning of a new block). Assume there are *500* distinct values of DEPTCODE, and that the EMPLOYEE records are evenly distributed among these values. Calculate:

a) The index blocking factor (bfr_i).
b) The number of first-level index entries (r1) and the number of first-level index blocks (b1).
c) The number of levels needed (x) if we make it a multi-level index.
d) The total number of blocks required by the multi-level index (b_i).
e) The number of block accesses needed to search for and retrieve all records in the file having a specific DEPTCODE value using the clustering index (assume that multiple blocks in a cluster are either contiguous or linked by pointers).

Ans:
a) the index blocking factor (bfr_i).
Index record size R_i = (V DeptCode + P) = (15 + 6) = 21 bytes
bfr_i = fo = floor(B/R_i) = floor(1024/21) = **48**
b) the number of first-level index entries (r1) and the number of first-level index blocks (b1).
r1 = number of distinct Department_code values= **500** entries
b1 = ceiling(r1 /bfr_i) = ceiling(500/48) = **11** blocks
c) the number of levels needed (x) if we make it a multi-level index.
We can calculate the number of levels as follows:
r2 = number of 1st-level index blocks b1 = 11 entries
b2 = ceiling(r2 /bfr_i) = ceiling(11/48) = 1 block;
Hence, the index has x = **2** levels
d) the total number of blocks required by the multi-level index (b_i).
b_i = b1 + b2 = 11 + 1 = **12** blocks
e) the number of block accesses needed to search for and retrieve all records in the file having a specific DEPTCODE value using the clustering index (assume that multiple blocks in a cluster are either contiguous or linked by pointers).
Number of block accesses to search for the first block in the cluster of blocks = x + 1 = 2 + 1 = 3
The s=2000 records are clustered in ceiling(s/bfr) = ceiling(2000/8) = 250 blocks.
Hence, total block accesses needed on average to retrieve all the records with a given DeptCode= x + 250 = **252** block accesses

Q3. *(6 points)* Suppose you are building an extensible hash index on a table of 100,000 rows. Key values are 8 bytes, a pointer (block/record) to a row is 8 bytes, and a disk block is 2048 bytes. Assume all keys are distinct.

a) What is the (lowest possible) global depth? Provide valid reasons.

b) What is the average occupancy of a bucket, assuming all buckets have a local depth equal to the global depth from part (a)? Justify your answer.

Ans:

a) Bucket entries will be key/pointer pairs, so 16 bytes each. Floor(2048/16) = 128 entries / bucket. 100,000/128 = at least 782 buckets needed. Since the directory is always a power of 2 size, it will have at least 2^{10} = 1024 entries, so the global depth is 10.

b) If all buckets have local depth equal to global depth, then every pointer in the directory points to a unique bucket. Thus, there are 1024 buckets. 1024 * 128 = capacity of 131,072. 100,000/131,072 ~= 76.3% occupancy.

RollNo: ______________ **Name: ________________________**

Q4. *(2+3+3= 8 points)* Assume that you have just built a dense B^+-tree index on a heap (unordered) file containing 1,000,000 records. The key field for this B^+-tree index is a 40-byte string, and it is a candidate key. Pointers (Record/block) are 10-byte values. The size of one disk page is 1000 bytes. The index was built in using the bulk-loading algorithm, and the nodes at each level were filled up as much as possible.

a. How many levels does the resulting tree have?

b. For each level of the tree, how many nodes are at that level?

c. How many levels would the resulting tree have with all pages 70 percent full?

Ans: see note book.

order p = 20; (p * 10) + ((p - 1) * 40) <= 1000

order p_{leaf} = 19; (p_{leaf} * (40+10)) + 10 <= 1000

b1= ceiling(1,000,000/19) = 52,632

fo = 20

b2= ceiling(52632/20)= 2,632

b3= ceiling(2632/20)= 132

b4= ceiling(132/20)= 7

b5= 1

a. x=5; OR x = ceiling($\log_{fo}$ (b1)) + 1 = ceiling($\log_{20}$ 52632) + 1 = 4 + 1 = 5 levels

b. Lev1(b1)=52632, Lev2(b2)=2632, Lev3(b3)=132, Lev4(b4)=7, Lev5(b5)=1

Avg no of keys in leaf nodes= ceiling(0.70*p_{leaf}) =.70* 19 = 14

Avg fo of internal node= fo = ceiling(.70*p) = .70*20 = 14

b1= ceiling(1,000,000/14) = 71,429

b2= ceiling(71,429/14)= 5103

b3= ceiling(5103/14)= 365

b4= ceiling(365/14)= 27

b5= ceiling(27/14)= 2

b6= 1

c. x=6 OR x = ceiling($\log_{fo}$ (b1)) + 1 = ceiling($\log_{14}$ 71429) + 1 = 5 + 1 = 6 levels