Assignment 3

P=6 bytes
P_R = 7 bytes
Record Length = 30+9+9+40+9+8+1+4+4+1 = R = 115 bytes

Part Number	**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: 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.	Blocking factor(BFR) = B/R =floor(4096/115) = **35** Number of blocks required = ceil(r/bfr) = 10,000,000/35 = **285715** i) **Index bfr_i** Index record size = R_i = (V_{SSN}+P) = 9+6 = 15 bytes Index blocking factor =B_{fri} = B/ R_i **= 4096/15 = 273** **ii) Number of First Level index Entries and block:** First Level Entries = R1 = ceil(10,000,000/bfr) = **285,715** First Level Index Blocks = B1 = R1/ B_{fri} = **285,715**/273 = **1047** **iii) Multilevel Index** b1 = 1047 b2 = ceil(1047/273) = 4 b3 = ceil(8/271) = 1 As third level has only one block, so **X = 3** **iv) Total Blocks Required** bi = b1+b2+b3 = 1047+ 4 + 1 = **1052** **v) No. of Block Access Needed** As X = 3, so block access = X+1 = 3+1 = **4**	Blocking factor(BFR) = B/R =floor(8192/115) = **71** Number of blocks required = ceil(r/bfr) = 10,000,000,000/71 = **140845071** i) **Index bfr_i** Index record size = R_i = (V_{SSN}+P) = 9+6 = 15 bytes Index blocking factor = B/ R_i **=** B_{fri} = **8192/15 = 546** **ii) Number of First Level index Entries and block:** First Level Entries = R1 = ceil(10,000,000,000/ bfr) = **140,845,071** First Level Index Blocks = B1 = Ri/ B_{fri} = 140,845,071/546= **257959** **iii) Multilevel Index** b1 = 257959 b2 = ceil(257959/546) = 473 b3 = ceil(473/546) = 1 As third level has only one block, so **X = 3** **iv) Total Blocks Required** bi = b1+b2+b3 = 257959+ 473+ 1 = **258433** **v) No. of Block Access Needed** As X = 3, so block access = X+1 = 3+1 = **4**

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.	Blocking factor(BFR) = B/R =floor(4096/115) = **35** i) **Index bfr$_i$** Index record size = R_i = (V_{SSN}+P) = 9+6 = 15 bytes Index blocking factor =B_{fri} = B/ R_i **= 4096/15 = 273** **ii) Number of First Level index Entries and block:** First Level Entries = R1 = 10,000,000 First Level Index Blocks = B1 = ceil(R1/B_{fri}) = ceil(10,000,000/273) = **36,631** **iii) Multilevel Index** b1 = 36631 b2 = ceil(36631/273) = 135 b3 = ceil(135/271) = 1 As third level has only one block, so **X = 3** **iv) Total Blocks Required** bi = b1+b2+b3 = 36631+ 135 + 1 = **36767** **v) No. of Block Access Needed** As X = 3, so block access = X+1 = 3+1 = **4**	Blocking factor(BFR) = B/R =floor(8192/115) = **71** i) **Index bfr$_i$** Index record size = R_i = (V_{SSN}+P) = 9+6 = 15 bytes Index blocking factor = B/ R_i **=** B_{fri} = **8192/15 = 546** **ii) Number of First Level index Entries and block:** First Level Entries = R1 = 10,000,000,000 First Level Index Blocks = B1 = ceil(R1/ B_{fri}) = ceil(10,000,000,000/546) = **183,150,19.** **iii) Multilevel Index** b1 = 18315019 b2 = ceil(18315019/546) = 33544 b3 = ceil(33544/546) = 62 b4 = ceil(62/546) = 1 As fourth level has only one block, so **X = 4** **iv) Total Blocks Required** bi = b1+b2+b3 = 18315019+ 33544+ 62+ 1 = **183,486,26** **v) No. of Block Access Needed** As X = 3, so block access = X+1 = 4+1 = **5**
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	i) **Index bfr$_i$** Index record size = R1 = (V_{SSN}+P) = 9+6 = 15 bytes Index blocking factor =B_{fri} = B/ R1 **= 4096/15 = 273** **ii) Number of blocks needed by level of indirection:** As, there are 20,000 distinct values and Pr = 7, so total bytes to store record	i) **Index bfr$_i$** Index record size = R1 = (V_{SSN}+P) = 9+6 = 15 bytes Index blocking factor = B/ R1 **=** B_{fri} = **8192/15 = 546** **ii) Number of blocks needed by level of indirection:** As, there are 20,000 distinct values and Pr = 7, so total bytes to store record pointer

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.	pointer for 1 department = 10,000,000/20,000 = 500 Now, 500*7 = 3500<4096, so it can fit in 1 block, so no. of blocks for **level of indirection = 20,000** **iii) Number of First Level index Entries and block:** r1 = distinct departments = 20,000 b1 = ceil(r1/BFRi) = 20,000/273 = **74** **iv) Multilevel Index** b2 = ceil(74/273) = 1 As second level has only one block, so **X = 2** **v) Total Blocks Required** bi = b1+b2+level of indirection = 74+ 1 + 20,000 = **20,075** **vi) No. of Block Access Needed** As X = 2, so block access = X+1+selectivity = 2+1+500 = **503**	for 1 department = 10,000,000,000/20,000 = 500,000 Now, 500,000*7 = 3500,000>8192, so it can't fit in 1 block. We will need 3500,000/8192 times blocks. So no. of blocks for **level of indirection = 20,000*428 = 8,560,000** **iii) Number of First Level index Entries and block:** r1 = distinct departments = 20,000 b1 = ceil(r1/BFRi) = 20,000/546 = **37** **iv) Multilevel Index** b2 = ceil(37/546) = 1 As second level has only one block, so **X = 2** **v) Total Blocks Required** bi = b1+b2+level of indirection = 37+ 1 + 8,560,000= **8,560,038** **vi) No. of Block Access Needed** As X = 2, so block access = X+1+selectivity = 2+1+500,000 = **500,003**
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	Blocking factor(BFR) = B/R =floor(4096/115) = **35** i) **Index bfr$_i$** Index record size = R_i = (V_{SSN}+P) = 9+6 = 15 bytes Index blocking factor =B_{fri} = B/ R_i **= 4096/15 = 273** **ii) Number of First Level index Entries and block:** First Level Entries = Ri = **20,000** First Level Index Blocks = B1 = ceil(Ri/B_{fri}) = ceil(20,000/273) = **74**	Blocking factor(BFR) = B/R =floor(8192/115) = **71** i) **Index bfr$_i$** Index record size = R_i = (V_{SSN}+P) = 9+6 = 15 bytes Index blocking factor = B/ R_i **=** B_{fri} = **8192/15 = 546** **ii) Number of First Level index Entries and block:** First Level Entries = Ri = **20,000** First Level Index Blocks = B1 = ceil(Ri/ B_{fri}) = ceil(20,000/546) = **37.**

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)	**iii) Multilevel Index** b2 = ceil(74/273) = 1 As second level has only one block, so **X = 2** **iv) Total Blocks Required** bi = b1+b2 = 74+ 1 = **75** **v) No. of Block Access Needed** Number of block accesses to search for the first block in the cluster of blocks **=x+1=2+1=3** The **74** records are clustered in ceiling(500/bfr) = ceiling(500/35) = **15** blocks. Hence, total block accesses needed on average to retrieve all the records with a given Department code = **x + 15 = 2 + 15 = 17 block accesses**	**iii) Multilevel Index** b2 = ceil(37/546) = 1 As second level has only one block, so **X = 2** **iv) Total Blocks Required** bi = b1+b2 = 37+1 = **38** **v) No. of Block Access Needed** Number of block accesses to search for the first block in the cluster of blocks **=x+1=2+1=3** The **37** records are clustered in ceiling(37/bfr) = ceiling(500,000/71) = **7043** block. Hence, total block accesses needed on average to retrieve all the records with a given Department code = **x + 7043 = 2 + 7043 = 7045 block accesses**
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	**i) the orders P and P_{leaf} of the B+-tree:** (p*P)+((p-1)*VSSN)<B: (p*6)+((p-1)*9)<4096 **15p < 4105** **P = 273** (**P_{leaf}** *(VSSN +PR))+P<B, (**P_{leaf}** * (9+7)) + 6 < 4096 16 **P_{leaf} <** 4090 **P_{leaf} = 255** **ii) the number of leaf-level blocks needed if blocks are approximately 69% full**	**i) the orders P and P_{leaf} of the B+-tree:** (p*P)+((p-1)*VSSN)<B: (p*6)+((p-1)*9)<8192 15p<8201 **P=546** (**P_{leaf}** *(VSSN +PR))+P<B, (**P_{leaf}** * (9+7)) + 6 < 8192 16 **P_{leaf} <** 8106 **P_{leaf} = 511** **ii) the number of leaf-level blocks needed if blocks are approximately 69% full** The average number of key values in a leaf node is 0.69* **P_{leaf}** = 0.69*511 = 352.59

value--using the B^+-tree.	The average number of key values in a leaf node is 0.69* P_{leaf} = 0.69*255 = 175.95 If we round this up for convenience, we get **176** key values (and **176** record pointers) per leaf node. Since the file has 10,000,000 records and hence 10,000,000 values of SSN, the number of leaf-level nodes (blocks) needed is b1 = ceiling(10,000,000 /176) = **56819 blocks.** **iii) If Internal nodes are 69% full:** **Fo** = ceiling(0.69*p) = ceiling(0.69***273**) = **189** **b2** = ceiling(b1 /fo) = ceiling(**56819** /**189**) =**301** **b3** = ceiling(b2 /fo) = ceiling(301/189)= **2** **b4** = ceiling(b3 /fo) = ceiling(2/189)= **1** Since the 4th level has only one block, the tree has **x = 4 levels** (counting the leaf level). x = ceiling(Log(Fo) (b1)) + 1 = 3 + 1 = **4 levels** **iv) Total Blocks Required** bi = b1+b2+b3+b4 = 56819+301+2+1 = **57123** **v) No. of Block Access Needed** **=X+1=4+1=5**	If we round this up for convenience, we get **353** key values (and **353** record pointers) per leaf node. Since the file has 10,000,000,000 records and hence 10,000,000,000 values of SSN, the number of leaf-level nodes (blocks) needed is b1 = ceiling(10,000,000,000 /**353**) = **28,328,612 blocks.** **iii) If Internal nodes are 69% full:** **Fo** = ceiling(0.69*p) = ceiling(0.69***546**) = **377** **b2** = ceiling(b1 /fo) = ceiling(**28,328,612** **/377**) =**75143** **b3** = ceiling(b2 /fo) = ceiling(**75143**/**377**)= **200** **b4** = ceiling(b3 /fo) = ceiling(**200**/**377**) = **1** Since the 4th level has only one block, the tree has **x = 4 levels** (counting the leaf level). x = ceiling(Log(Fo) (b1)) + 1 = 3 + 1 = **4 levels** **iv) Total Blocks Required** bi = b1+b2+b3+b4 = **28,328,612** +**75143**+**200**+**1** = **28,403,956** **v) No. of Block Access Needed** **= X+1 = 4+1 =5**
F: Repeat (part e), but for a B-tree *rather than for a* B^+-tree. Compare your results for the	**i) the orders P and P_{leaf} of the B+-tree:** (p*P)+((p-1)* (VSSN+Pr))<B:	**i) the orders P and P_{leaf} of the B+-tree:** (p*P)+((p-1)* (VSSN+Pr))<B:

B-tree and for the B^+-tree.	(p*6)+((p-1)* (9+7)) <4096 **22P < 4112** **P = 186** **ii) the number of leaf-level blocks needed if blocks are approximately 69% full** The average number of key values in a leaf node is 0.69* **P** = 0.69*186 = 128.34 If we round this up for convenience, we get **129** key values (and **129** record pointers) per leaf node. Since the file has 10,000,000 records and hence 10,000,000 values of SSN, the number of leaf-level nodes (blocks) needed is b1 = ceiling(10,000,000 /**129**) = **77520 blocks.** **iii) If Internal nodes are 69% full:** **Fo** = ceiling(0.69*p) = ceiling(0.69*186) = **129** **b2** = ceiling(b1 /fo) = ceiling(**77520** /**129**) =**601** **b3** = ceiling(b2 /fo) = ceiling(**601** /**129**)= **5** **b4** = ceiling(b3 /fo) = ceiling(5/129)= **1** Since the 4th level has only one block, the tree has **x = 4 levels** (counting the leaf level). **x** = ceiling(Log(Fo) (b1)) + 1 = 3 + 1 = **4 levels** **iv) Total Blocks Required**	(p*6)+((p-1)* (9+7)) <8192 **22P < 8208** **P = 373** **ii) the number of leaf-level blocks needed if blocks are approximately 69% full** The average number of key values in a leaf node is 0.69* **P** = 0.69*373 = 257.37 If we round this up for convenience, we get **258** key values (and **258** record pointers) per leaf node. Since the file has 10,000,000,000 records and hence 10,000,000,000 values of SSN, the number of leaf-level nodes (blocks) needed is b1 = ceiling(10,000,000,000 /**258**) = **38,759,690 blocks.** **iii) If Internal nodes are 69% full:** **Fo** = ceiling(0.69*p) = ceiling(0.69*373) = **258** **b2** = ceiling(b1 /fo) = ceiling(**38,759,690 /258**) = **150232** **b3** = ceiling(b2 /fo) = ceiling(**150232/258**)= **583** **b4** = ceiling(b3 /fo) = ceiling(**583/258**) = **3** **b5** = ceiling(b3 /fo) = ceiling(**3/258**) = **1** Since the 5th level has only one block, the tree has **x = 5 levels** (counting the leaf level). **x** = ceiling(Log(Fo) (b1)) + 1 = 4 + 1 = **5 levels**

	bi = b1+b2+b3+b4 = 77520 +601 +5+1 = **78127 blocks.** **v) No. of Block Access Needed** **=X+1=4+1=5**	**iv) Total Blocks Required** bi = b1+b2+b3+b4 = 38,759,690 +150232+583+3+1 = **38,910,509 blocks** **v) No. of Block Access Needed** **= X+1 = 5+1 =6** As B-Tree requires more block access compared to B+ Tree, therefore for this part, it is more costly as compared to 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)
Block Size = 1Kb = 1024 bytes
Pointer = 12 byte
Bhead = 56 bytes
Let us assume that each B+ Tree has n pointers and n-1 keys.
So, Formula = (p*(n-1))+((n)*Pointer + Bhead) <= B,
Here p = 8 bytes, so

- ⇨ (8 * (n-1) + (12n) + 56) <= 1024
- ⇨ 8n + 12n <= 976
- ⇨ 20n <= 976
- ⇨ n <= 48.8
- ⇨ Therefore, n<=48
- ⇨ **So n=47**

Now, As leaf node of B+ Tree can hold record pointers = 47*48*48 = **108288**.
So maximum number of records we can index = **108,288.**

b. 3 Level B-tree index (including the root level)
Let B tree have n index pointers, n-1 keys and record pointers, therefore,
Formula = 8*(n-1) + 12*(2n-1) + 56 <= 1024

- ⇨ 8n – 8 + 24n -12 + 56 <= 1024
- ⇨ 32n <= 988
- ⇨ n <= 30

so n=29.
1^{st} Level B-Tree can hold = 29
2^{st} Level B-Tree can hold = 29*30 = 870
3^{st} Level B-Tree can hold = 29*30*30 = 26100
Total = 29 + 870 +26100 = 26,999
So maximum number of records we can index = **26,999.**

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:

Data blocks = 5000

1. ***SELECT * FROM R WHERE A = 777;***

Number of block Fetches = X+1 = **5**

2. ***SELECT C FROMRWHERE A=111 AND B=3;***

Number of block Fetches = = **X+1 = 5**

3. ***SELECT* FROMRWHEREA=111 OR A=3;***

Number of block Fetches = 2*(X+1) = 2(5) =**10**

4. ***SELECT * FROM R WHERE A > 100;***

= 4+5000=5004 Assuming at worst case all values are greater than 100

5. ***SELECT COUNT(*) FROM R WHERE A > 100;***

Number of block Fetches = X = 4