Name: _____

Bitmaps	
B-trees	
Extensible hashing	
Outerjoins	
Algebra and estimation	

You may consult the textbook, my slides and any notes you have.

You have 70 minutes. Write the answers on the exam.

Bitmap Index (7)

For the example table discussed in the class, provide the remaining three *compressed* bitmap values

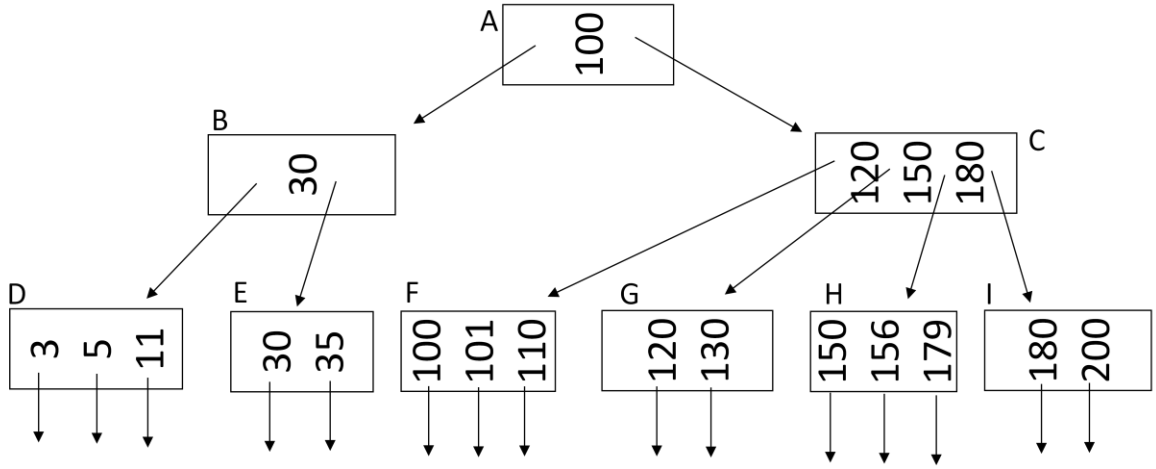
Key	Bitmap	Compressed
Toys	00011010	10110001
PCs	00100100	
Pens	01000001	
Suits	10000000	

B+ Tree Index (7)

Draw the tree that will result from the insertion of a record with key 115. **NOT JUST ANY TREE THAT HAS THESE DATA.**

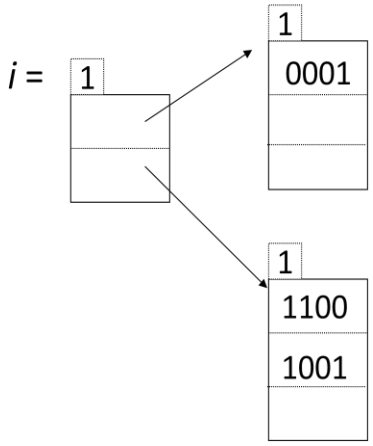
To save yourself from some drawing, do not redraw subtrees that are not modified. Just put the label of the subtree in the picture. For example, the subtree C stands for the node marked with C and all its children.

n=3



Extensible hash index (10)

Consider an extensible hash index that uses 4 bits and each block of the index contains up to 3 entries. Initially the index indexes three keys with respective hash values 0001, 1100, 1001. As you see, we will use the starting bits in this exercise. As we did in the slides' example also, instead of showing the actual key values we show just the hashed values of the keys.



Show the state of the extensible index after keys with hash values 1010, 0111, 1000 and 1011 have been inserted. Order of keys within the block does not matter.

Outerjoins (16)

WRITE EXAMPLE WITH A1, A2, B, C1, C2

Consider the extended projection operator we discussed in the lectures and assume that a function that may be used in the extended projection is the null(), which produces the special value NULL. For example, assuming the following R

A	**B**
1	2
1	2
3	4

the result of $\pi_{A,B,null()->F}R$ is

A	**B**	**F**
1	2	NULL
1	2	NULL
3	4	NULL

Next consider the natural left outerjoin operator **LOUT**.

Given two relations R and S where B is the set of attributes that are common to R and S, A is the set of attributes that appear only in R and C is the set of attributes that appear only in S, the R **LOUT** S is defined as follows:

- its set of attributes is the concatenation of A, B and C attributes
- for every pair of tuples r from R and s from S such that r and s have equal values in the attributes of B, the result has a tuple with the A, B and C values of r and s.
- for every tuple r of R such that there is no tuple of S that has equal values with r on the common attributes B, the result has a tuple with the A and B values of r and NULL values in C

Provide and algebraic definition for R **LOUT** S using operators we have seen in the notes, including the generalization of projection described above.

Assume relations R, S and T. The notation c(A) refers to a condition that refers to a list of attributes A. As is the case with SQL, any condition that is evaluated on an attribute whose value is NULL returns false, for the particular tuple.
Declare true or false each of the following. If the answer is “no", also provide counterexample.

- $\sigma_{C(A)}$(R **LOUT** S) = ($\sigma_{C(A)}$R) **LOUT** S, where R has a list of attributes A
- $\sigma_{C(C)}$(R **LOUT** S) = R **LOUT** ($\sigma_{C(C)}$ S), where S has a list of attributes C but R has no attribute of the list C.
- R **LOUT** (S **LOUT** T) = (R **LOUT** S) **LOUT** T, where all of R, S and T have a single common attribute A and no pair of R, S and T has a common attribute other than A.
- δ (R **LOUT** S**) =** (δR) **LOUT** S

Algebra & estimation (20)

Consider three relations R(A;B;C), S(A;D), W(B;E) and the query

```
SELECT *
FROM R, S, W
WHERE R.A=S.A AND R.B=W.B AND R.C=1
```

Consider an optimizer that produces all join expressions, where the selection $\sigma_{R.C=1}$ is pushed down and applied directly on the table R.

Plans cannot have cartesian products or trivial natural joins that are equivalent to cartesian products.

Write all possible logical query plans (i.e., algebraic expressions) that this optimizer will produce but do not write twice expressions that can be derived from each other just by using the commutativity of the natural join.

Size Estimation

Assume that the optimizer selects the best logical query plan by estimating the size of each intermediate result (i.e., of each subexpression) and selecting the plan that has the smallest sum of intermediate result sizes. For each of the logical query plans you provided earlier, calculate the size of each intermediate result and decide which is the best plan. Note that you do not have to estimate the size of the end result since it should always be the same.

Use the following data for your estimates. Write whether your estimate is the precise number, i.e., whether the number will be the same for all possible databases that conform to the following characteristics.

- B is a key of R and W.B is a non-null foreign key that references R.B
- A is a key of S and R.A is a non-null foreign key that references S.A
- $T(R) = 10^6$
- $T(S) = 10^5$
- $T(W) = 10^3$
- $V(R.C) = 10^2$