## Question 1: Trees (Part 1) . . . . . [330 points]

(i) **[10 points] Table Indexes:**
Define a table index. List the key and value of an in-memory hash index. List the key and value of an on-disk tree index.

(ii) **[10 points] Table Indexes:**
Are indexes base or derived data structures? Why?

(iii) **[10 points] Table Indexes:**
List a benefit and a limitation of building an index.

(iv) **[10 points] Table Indexes:**
Do indices accelerate read-intensive workloads or write-intensive workloads? Why?

(v) **[10 points] Table Indexes:**
Do indices benefit OLTP workloads or OLAP workloads more? Why?

(vi) **[10 points] B+Tree:**
How is a B+Tree optimized for disk storage as opposed to a hash table?

(vii) **[10 points] B+Tree:**
List the keys and values of a leaf node. List the keys and values of an inner node.

(viii) **[10 points] B+Tree:**
Explain the purpose of: (1) child pointers, (2) parent pointers, and (3) sibling pointers.

(ix) **[10 points] B+Tree:**
Are the pointers in a B+Tree node logical or physical pointers? Justify your answer.

(x) **[10 points] B+Tree:**
Distinguish between primary and secondary indexes with respect to the values stored in the leaf nodes.

(xi) **[10 points] B+Tree:**
Distinguish between BTree and B+Tree.

(xii) **[10 points] Node Split:**
Explain the node split operation in a B+Tree with an example.

(xiii) **[10 points] Node Merge:**
Explain the node merge operation in a B+Tree with an example.

(xiv) **[10 points] Operations:**
Explain how `FIND` operation works in a B+Tree.

(xv) **[10 points] Operations:**
Explain how `INSERT` operation works in a B+Tree.

(xvi) **[10 points] Operations:**
Explain how `DELETE` operation works in a B+Tree.

(xvii) **[10 points] Data Organization:**
Distinguish between heap-organized and index-organized storage.

Question 1 continues…

(xviii) **[10 points] Data Organization:**
Distinguish between clustered and unclustered indexes. How are they connected to heap-organized storage? How are they connected to index-organized storage?

(xix) **[10 points] Unclustered Index:**
Distinguish between storing a clustered index pointer or a tuple pointer as the value in an unclustered index.

(xx) **[10 points] Filtering Tuples:**
When can a B+Tree index be used for filtering tuples? Illustrate with an example.

(xxi) **[10 points] Filtering Tuples:**
When can a B+Tree index **not** be used for filtering tuples? Illustrate with an example.

(xxii) **[10 points] B+Tree Design Decisions:**
How does the node size vary based on the device latency? Why?

(xxiii) **[10 points] B+Tree Design Decisions:**
How does the node size vary based on the workload? Why?

(xxiv) **[10 points] B+Tree Design Decisions:**
Distinguish between eager and lazy merge operations.

(xxv) **[10 points] B+Tree Design Decisions:**
Distinguish between these two techniques for storing variable length keys: (1) pointers and (2) key map.

(xxvi) **[10 points] B+Tree Design Decisions:**
Distinguish between these two techniques for handling duplicate keys: (1) duplicate keys and (2) value lists.

(xxvii) **[10 points] B+Tree Design Decisions:**
Distinguish between these three techniques for intra-node search: (1) linear search, (2) binary search, and (3) interpolation search.

(xxviii) **[10 points] B+Tree Design Decisions:**
Explain how interpolation search works with an example.

(xxix) **[10 points] B+Tree Design Decisions:**
When is interpolation search faster than binary search?

(xxx) **[10 points] B+Tree Optimizations:**
Explain the prefix compression optimization with an example.

(xxxi) **[10 points] B+Tree Optimizations:**
Explain the suffix truncation optimization with an example.

(xxxii) **[10 points] B+Tree Optimizations:**
Explain the bulk insert optimization with an example.

(xxxiii) **[10 points] B+Tree Optimizations:**
Explain the pointer swizzling optimization with an example. How is this optimization used in Leanstore?

End of Problem Set