Consider the B+ tree index of degree \( d = 2 \) shown in Figure 1. Instead of pointers to the data records in data files, the leaf nodes in this B+ tree contain data records. The insertion, deletion, and update algorithms for this B+ tree is the same as the ones discussed in the lecture.

1. Show the tree that would result from inserting a data entry with key 9 into this tree.

   **Solution:** The data entry with key 9 is inserted on the second leaf page. The resulting tree is shown in Figure 2.

2. Show the B+ tree that would result from inserting a data entry with key 3 into the original tree.

   **Solution:** The data entry with key 3 goes on the first leaf page \( F \). Since \( F \) can accommodate at most four data entries (\( d = 2 \)), \( F \) splits. The lowest data entry of the new leaf is given up to the ancestor which also splits. The result can be seen in Figure 3.

3. Show the B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the left sibling is checked for possible redistribution.

   **Solution:** The data entry with key 8 is deleted, resulting in a leaf page \( N \) with less than two data entries. The left sibling \( L \) is checked for redistribution. Since \( L \) has more than two data entries, the remaining keys are redistributed between \( L \) and \( N \), resulting in the tree in Figure 4.
Figure 3: The B+ tree after the changes in (2).

Figure 4: The B+ tree after the changes in (3).
4. Show the B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the right sibling is checked for possible redistribution.

**Solution:** As is part 3, the data entry with key 8 is deleted from the leaf page N. N’s right sibling R is checked for redistribution, but R has the minimum number of keys. Therefore the two siblings merge. The key in the ancestor which distinguished between the newly merged leaves is deleted. The resulting tree is shown in Figure 5.

5. Show the B+ tree that would result from starting with the original tree, inserting a data entry with key 46 and then deleting the data entry with key 52.

**Solution:** The data entry with key 46 can be inserted without any structural changes in the tree. But the removal of the data entry with key 52 causes its leaf page L to merge with a sibling (we chose the right sibling). This results in the removal of a key in the ancestor A of L and thereby lowering the number of keys on A below the minimum number of keys. Since the left sibling B of A has more than the minimum number of keys, redistribution between A and B takes place. The final tree is depicted in Figure 6.

6. Show the B+ tree that would result from deleting the data entry with key 91 from the original tree.

**Solution:** Deleting the data entry with key 91 causes a scenario similar to part 5. The result can be seen in Figure 7.
7. Show the B+ tree that would result from starting with the original tree, inserting a data entry with key 59, and then deleting the data entry with key 91.

**Solution:** The data entry with key 59 can be inserted without any structural changes in the tree. No sibling of the leaf page with the data entry with key 91 is affected by the insert. Therefore deleting the data entry with key 91 changes the tree in a way very similar to part 6. The result is depicted in Figure 8.

8. Show the B+ tree that would result from successively deleting the data entries with keys 32, 39, 41, 45, and 73 from the original tree.

**Solution:** Considering checking the right sibling for possible merging first, the successive deletion of the data entries with keys 32, 39, 41, 45 and 73 results in the tree shown in Figure 9.

---

### 2: B+ Tree Indexing

The degree of a B+ tree (d) can be a non-integer value. If the value of d is not integer, the minimum and maximum number of keys in the internal nodes will be floor(d) and floor(2d), respectively. The root is the exception and can have between 1 and floor(2d) keys. Consider the B+ tree shown in Figure 10, where the value of d is 1.5, so each internal node has between 1 and
3 keys.

1. Delete keys 13 and 17 from this B+ tree.

2. Insert keys 49, 51, 55 and 60 into this B+ tree.

You only need to show the final picture of the B+ tree.

**Solution:** The solution is shown in Figure 11.

### 3: B+ tree Indexing

Using B+ trees with degrees of two (2) whose keys are integer values, answer the following questions. A B+ tree with a single node has height of 1.

1. Give an example of a B+ tree whose height changes from 2 to 3 when the value 25 is inserted. Show your structure before and after the insertion.

2. Give an example of a B+ tree in which the deletion of the value 25 causes a merge of two nodes but without altering the height of the tree.
Solution:

1. One solution is a B+ tree has a root and five leaf nodes. The root node contains four keys of [10, 20, 30, 40]. The five leaf nodes, from left to right, contain [2,6], [10,13,16,17], [20,21,23,28], [31,32,36,38], and [43,54,69,87]. After inserting 25, the resulting B+ tree will have one root node, two internal nodes, and six leaf nodes. The root node contains a single key 23. The internal nodes, from left to right, contain [10,20] and [30,40]. The leaf nodes, from left to right, contain keys [2,6], [10,13,16,17], [20,21], [23,25,28], [31,32,36,38], and [43,54,69,87].

2. One solution is a B+ tree has a root and three leaf nodes. The root node contains four keys of [10, 20]. The three leaf nodes, from left to right, contain [2,6], [10,13,16], and [20,25]. After deleting 25, the resulting B+ tree will have one root node and two leaf nodes. The root node contains a single key of [10]. The two leaf nodes, from left to right, contain [2,6] and [10,13,16,20].

4: B+ tree Indexing

To reduce the number of I/O access in index search, each B+ tree node should fit in a block. Let the key value and record pointer for a B+ tree be 32 and 64 bytes, respectively. If the block size is 16384 bytes, what should be the minimum degree of the B+ tree?

Solution: $2d \times 32 + (2d + 1) \times 64 \leq 16384$. 