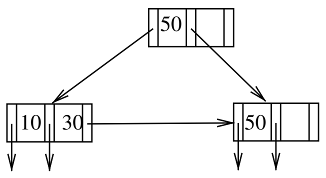
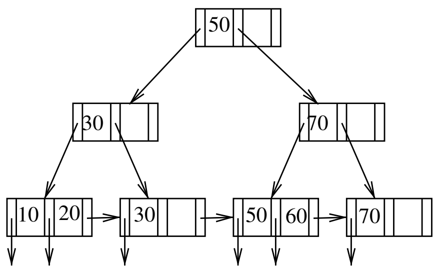
CS143: Database Systems
Answeres for Homework #4

2. Consider the following two B+trees for this problem.



(a)

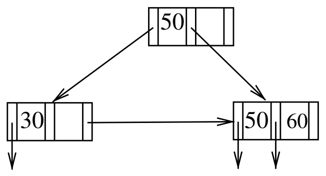


(b)

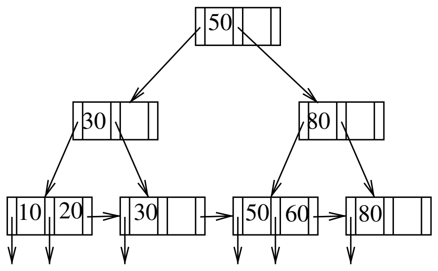
(a) Show the final B+tree structure after we insert 60, 20, and 80 into Figure (a) in the given order.

(b) Show the final B+tree structure after we delete 20, 10, and 70 from Figure (b) in the given order.

ANSWER:



(b)



(a)

3. Consider a B+tree that indexes 300 records. Assume that $n = 5$ for this B+tree (i.e., each node has at most 5 pointers), what is the minimum and maximum height (depth) of the tree? (A tree with only the root node has a height of 1.)

ANSWER:
Minimum 4. (maximum 4 record pointers per node at leaf. $\lceil 300/4 \rceil = 75$ leaf nodes are needed when full. maximum branching factor 5 at non-leaf nodes. $\lceil 75/5 \rceil = 15$ nodes are needed at level 2. $\lceil 15/5 \rceil = 3$ nodes are needed at level 3. One more level of root node that points to these three nodes.)

Maximum 5. (minimum 2 record pointers per node at leaf. $\lfloor 300/2 \rfloor = 150$ leaf nodes. minimum branching factor 3 at non-leaf nodes. $\lfloor 150/3 \rfloor = 50$ nodes at level 2. $\lfloor 50/3 \rfloor = 16$ nodes at level 3. $\lfloor 16/3 \rfloor = 5$ nodes at level 4. $\lfloor 5/3 \rfloor = 1$ nodes at level 5. Since there is only one node at level 5, this is the root node.)

4. Consider the following key values:

```
106, 115, 916, 0, 96, 126, 16, 15, 31
```

These keys are to be inserted in the above order into an (initially empty) extendible hash table. The hash function $h(n)$ for key n is $h(n) = n \bmod 256$; that is, the hash value is the remainder when the key value is divided by 256 (2^8). Thus, the hash value is an 8-bit value. Each block can hold 3 data items. Draw the extendible hash table after all data items are inserted. Show the keys themselves in the buckets, not the hash value. The bucket numbers are drawn from the bits at the high order end of the hash value. Be sure to indicate i for the directory, the number of hash value bits used. Also indicate i for each bucket, the number of hash function bits that are used for that bucket.

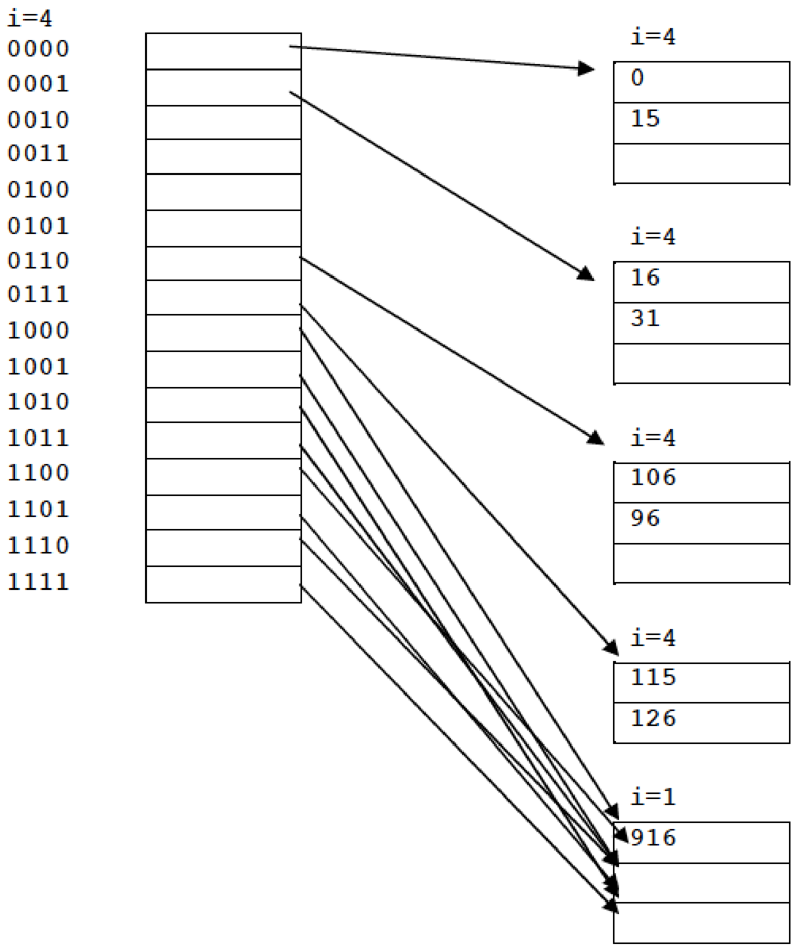
ANSWER:

```
1.
Data: 106, 115, 916, 0, 96, 126, 16, 15, 31

Keys:

H(106) = 106 = 01101010
H(115) = 115 = 01110011
H(916) = 148 = 10010100
H(0)   =   0 = 00000000
H(96)  =  96 = 01100000
H(126) = 126 = 01111110
H(16)  =  16 = 00010000
H(15)  =  15 = 00001111
H(31)  =  31 = 00011111

The cells with no pointers point to empty buckets
```



Problem 4. Answer Questions 1,2,3, and 4 below

Suppose you have 2 relations, $R(\overline{A}, B)$ and $S(\overline{B}, C)$, with the following characteristics:

- The size of one disk block is 1000 bytes.
- Attributes A, B are of length 10 bytes. Attribute C is of length 180 bytes. [50R tiuple per block]
- The tuples are not spanned across disk blocks [5 tuples per block]
- $|R| = 5,000$ (number of tuples of R) [100 blocks]
- $|S| = 500$ (number of tuples of S) [B+C= 90: 5 S tuples per block: 100 blocks]
- We have 30 blocks of memory buffer
- We use one disk block for one B+tree node
- Each pointer in a B+tree index (both a record pointer and a node pointer) uses 10 bytes.

1. (2 points) What is the minimum number of blocks needed to store R and S?

   Answer: R= 100 S=100:

2. (2 points) We want to construct a dense B+tree on attribute B of table S by scanning the S table. What is the maximum possible n for the B+tree given the above parameters?

   n: ____________________
   ANSWER:
   $10n + 10(n-1) \leq 1000$. `Therefore,` $n = 50$.

3. Assume the numbers computed in the previous problems. Assume that each node in the B+tree contains the minimum number of keys and pointers (as long as it is allowed in our parameter setting).

   (a) (4 points) How many nodes does the constructed B+tree have?

   ANSWER:
   `21. For leaf nodes, the minimum number of keys per node is 25. Therefore, we will have` $500/25 = 20$ `leaf nodes and one root node. Any answer with 500/(half of answer 2)+1 would be considered correct assuming that answer 2 is not too small.`

   (b) (4 points) How many disk IOs would be incurred during the construction of the B+tree on `S.B`? Assume that you use the main memory buffer in the most efficient way to minimize the number of disk IOs. *In your answer, please include the cost of reading the tuples from S and writing the constructed B+tree to the disk.*

   ANSWER:
   `121. For index construction, we need to scan the entire` S `table once.` S `is stored in 100 blocks, so 100 disk IOs for reading` S`. We also incur 21 disk IOs to write the B+tree. Any answer with S from answer 1 + answer 3(a) would be considered correct assuming that 3(a) is not too big.`

4. (4 points) How many disk IOs would have been incurred if we used block nested loop join? Is it worthwhile to construct an index on `S.B` to perform the join using the above algorithm considering the index construction overhead?

   ANSWER:
   `block nested loop join would have incurred` $100 + 4 \times 100 = 500$ `disk IOs. Even if we take the index construction overhead, the total number of disk IOs was 242. It was definitely worthwhile. Any answer with (1R + 4x1S) or (1R + 100x1S) would be considered correct.`