# Data Structures and Algorithms

# CSYE 6205

# Homework 6

# Due: February 25, 2019

Put all your work, code, compiled, and executable files and documentation into a zip file named Homework6.zip and submit it via the drop box on the blackboard by the END of due date. Put your name on all code files. There will be a short quiz on this homework.

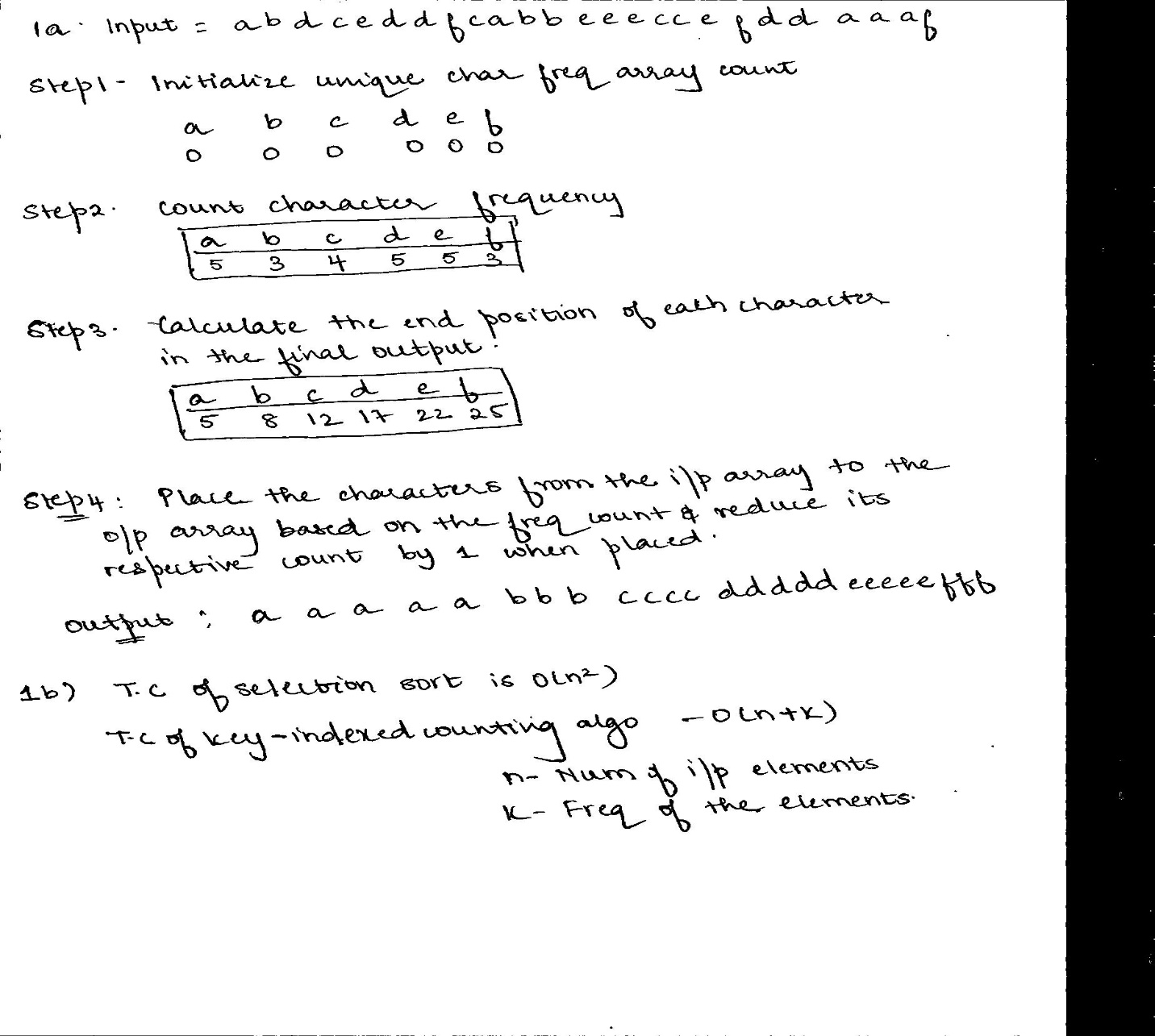
1. Consider this string : “abdceddfcabbeeeccefddaaaf”

a) Use key-indexed counting sort algorithm to sort the string. Show each step

and the results.

b) What is the running time complexity of the algorithm as compared to selection sort?

c) Write java code to sort the string using steps described in (a).



2. Consider Data: {9 30 -28 -25 -2 -3 6 12 15 19 25 26 33 36 39}

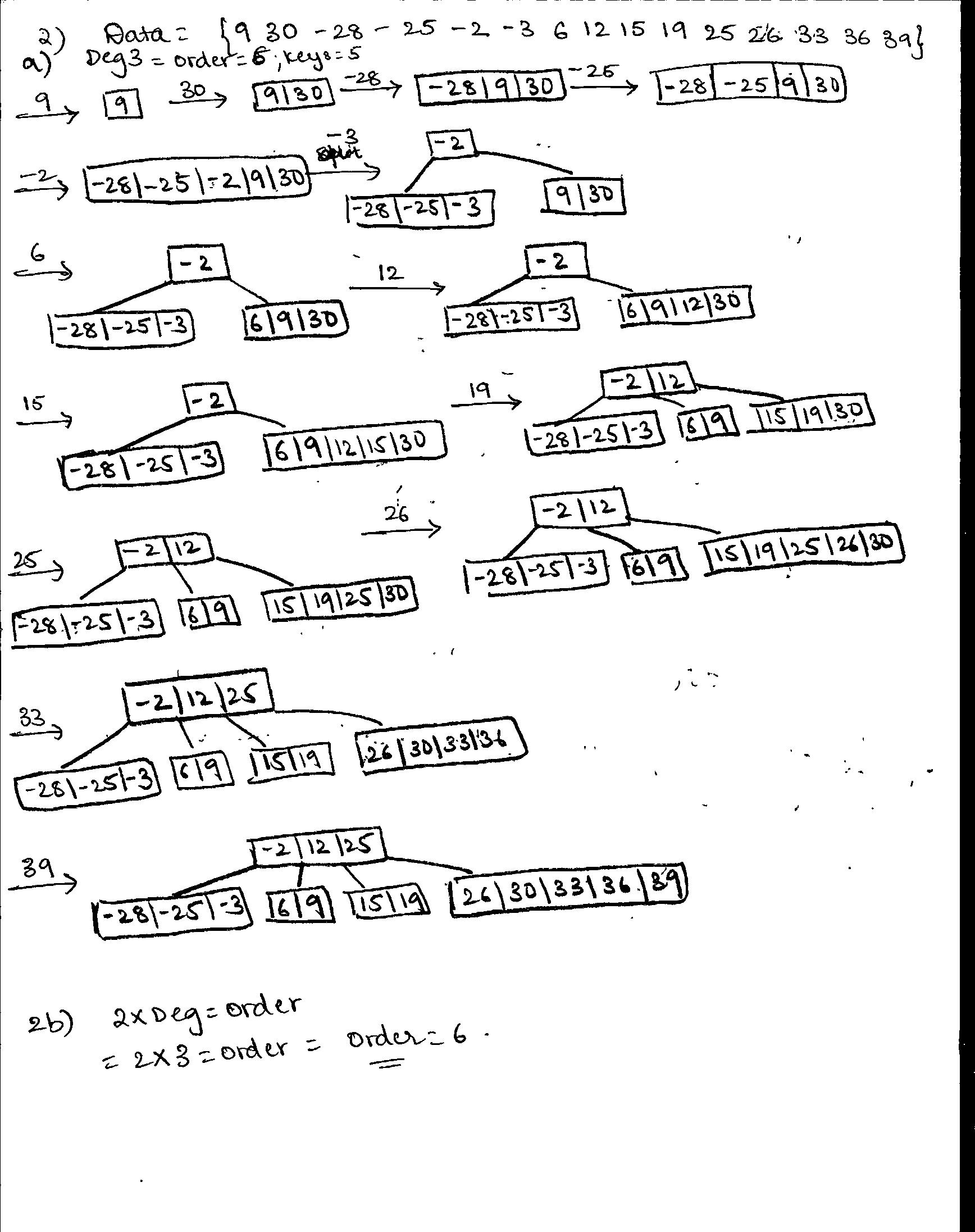
a) Build a btree with minimum degree of 3

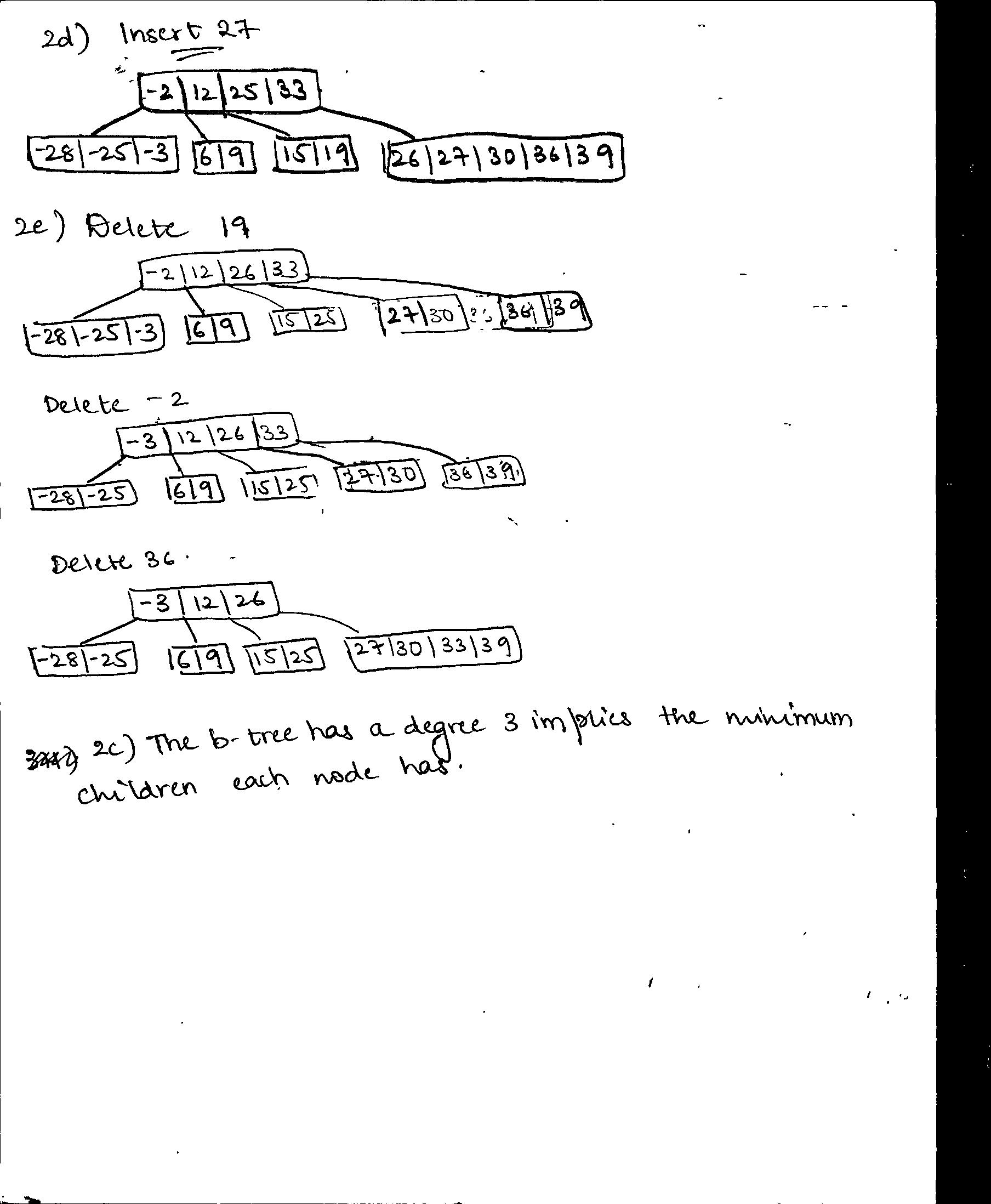
b) What is the order of this btree?

c) Why this is btree with minimum degree of 3?

d) Insert 27 into btree

e) Delete 19, -2, 36





3. Consider the following sequence of letters:

‘A’G’F’B’K’D’H’M'J'E'S'I'R'X'C'L'N'T'U'P'

a) Build BTree with order of t=4

b) What is minimum degree for this BTree?

c) Write Java code to Insert into BTree

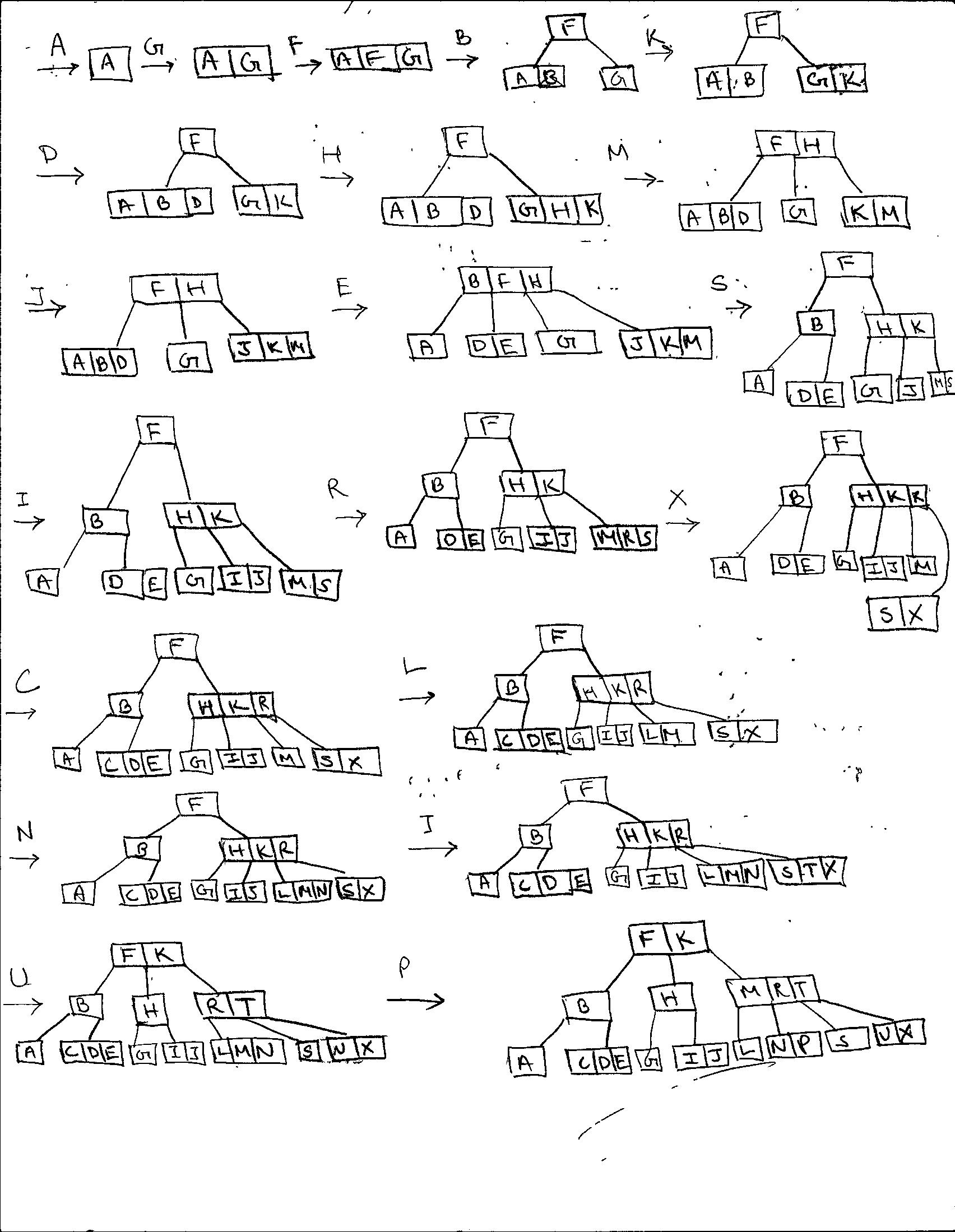
d) Consider 3-cases for deleting from B-tree,

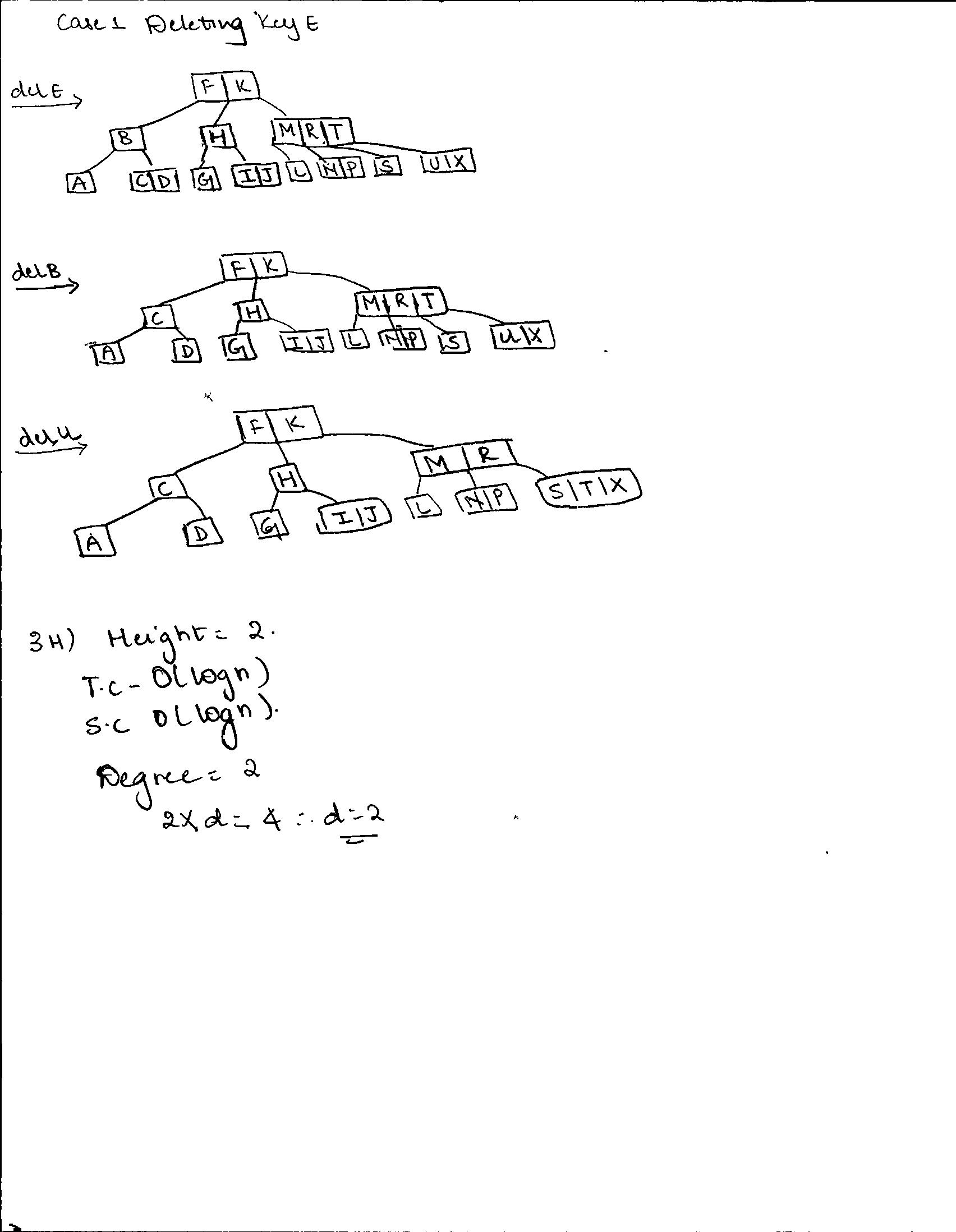
Select delete element for each of 3-cases to delete from BTree in (a)

e) Write Java code for all 3 deletion cases

f) Test Java code for (c) and (e) for the BTree you constructed in (a)

H) Discuss height, time and space complexity





4. Consider a disk with a sector size of 512 bytes, 1,000 tracks per surface, 100 sectors per track, five double-sided platters and a block/page size of 2,048 bytes. Suppose that the average seek time is 5 msec, the average rotational delay is 5 msec, and the transfer rate is 100 MB per second. Suppose that a file containing 1,000,000 records of 100 bytes each is to be stored on such a disk and that no record is allowed to span two blocks.

a) How many records fit onto a block? **20 records**

b) How many blocks are required to store the entire file? **50000** records

**Appendix-A DELETING from B-Tree**

1. x is a leaf and contains the key (it will have at least t keys). This is

trivial - just delete the key.

2. x is an internal node and contains the key. There are 3 subcases:

2a: predecessor child node has at least t keys

2b: successor child node has at least t keys

2c: neither predecessor nor successor child has t keys

3. x is an internal node, but doesn’t contain the key. Find the child subtree

of x that contains the key if it exists (call the child c). There are three subcases:

3a: child c has at least t keys. Simply recurse to c.

3b: child c has t − 1 keys and one of its siblings has t keys.

3c: child c and both siblings have t − 1 keys.

**Appendix-B INSERTING into B-Tree**

1: procedure B-Tree-Insert(Node x, Key k)

2: find i such that x.keys[i] > k or i >=numkeys(x)

3: if x is a leaf then

4: Insert k into x.keys at i

5: else

6: if x.child[i] is full then

7: Split x.child[i]

8: if k > x.key[i] then

9: i ← i + 1

10: end if

11: end if

12: B-Tree-Insert(x.child[i], k)

13: end if

14: end procedure