Mid-2 Marking Scheme

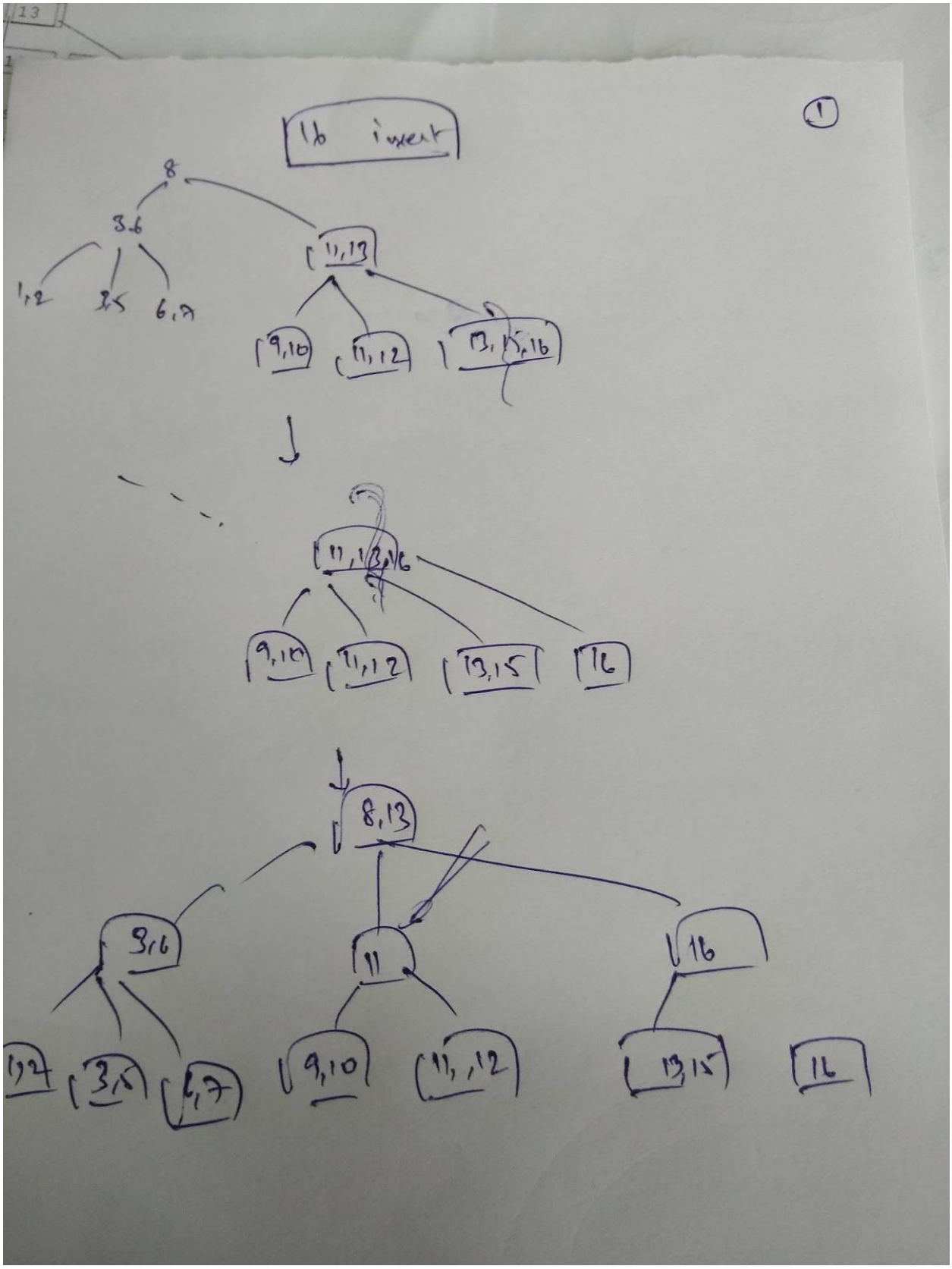
Question 1

(a). Look up all records in range 5 to 10 [1 Mark]
Compare with the intermediate node's data and go to the branch which contains 5 until leaf is reached, do a right traversal and print data till its <=10.

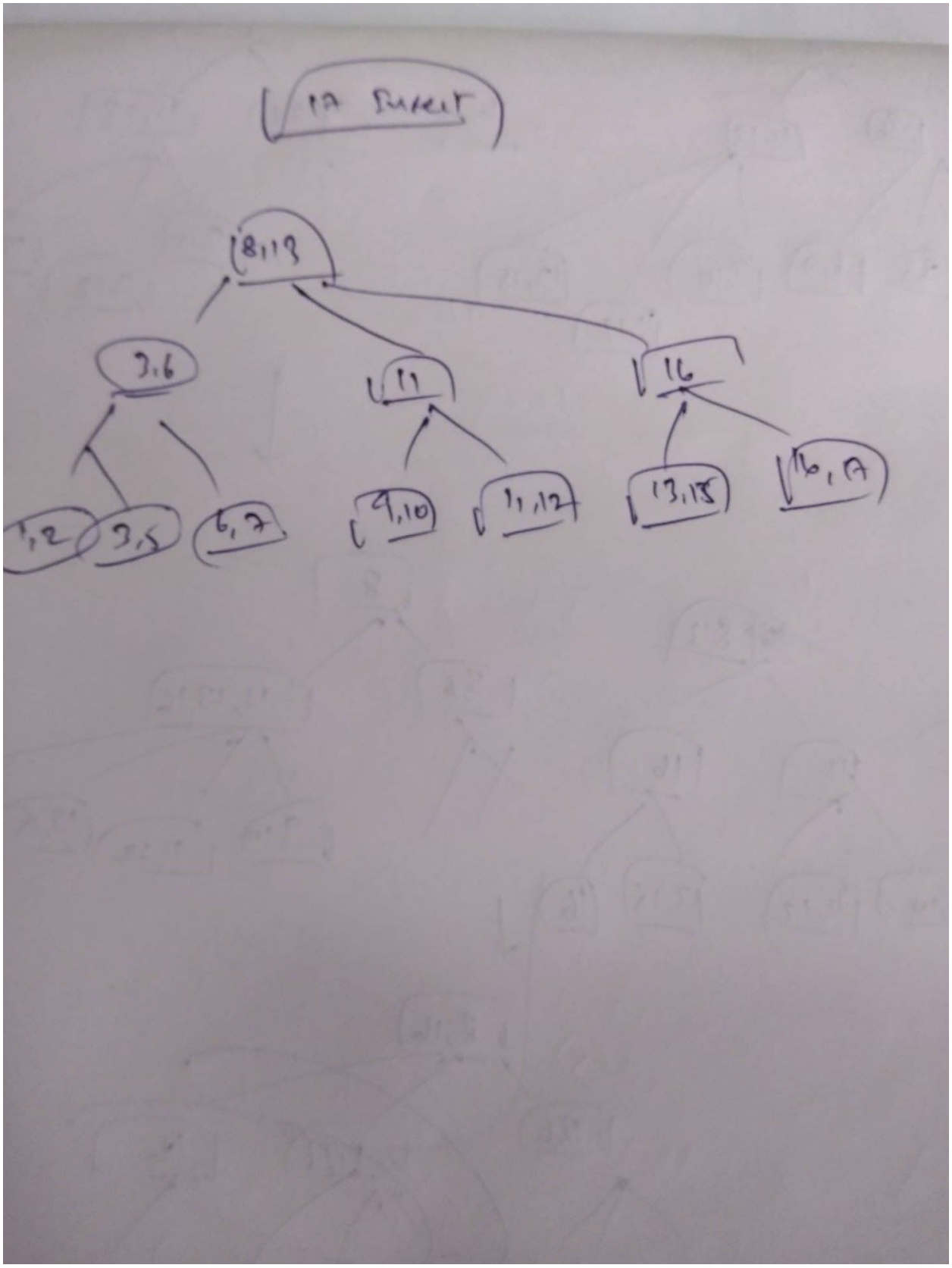
(b). Lookup all records less than 14 [1 Mark]
Go to leftmost leaf and iterate right and print till data is <14.

(c). Insert elements (16,17,18) [3 Marks]
(for 3 elements split can be in 1:2 or 2:1 fashion, considering both)
(while splitting intermediate nodes, mid element will be removed and pushed up, not copied)

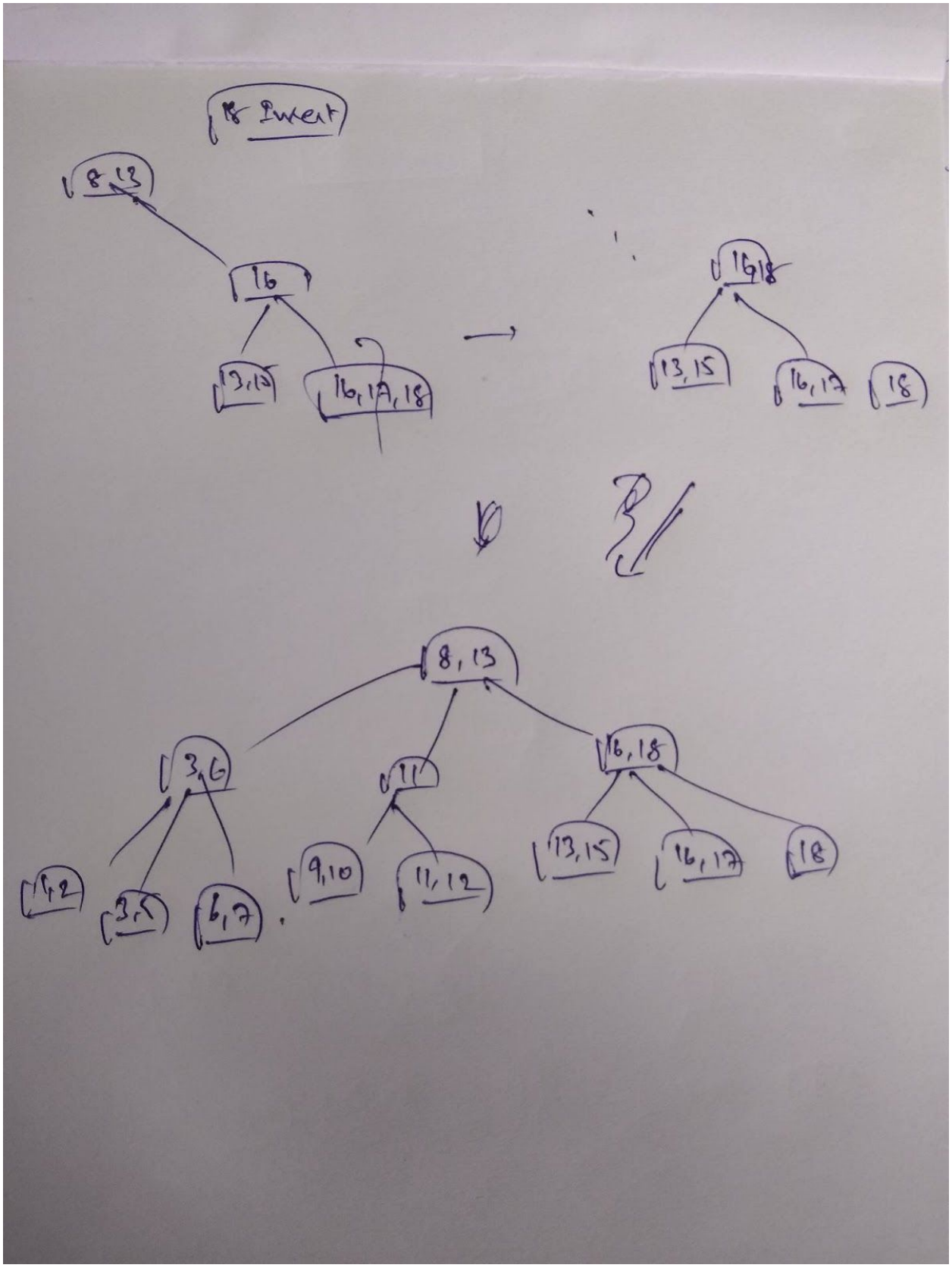
Insert 16


16 insert
1
8
3,6
1,2
6,7
11,13
9,10
11,12
13,15,16
11,13,16
9,10
11,12
13,15
16
8,13
3,6
11
16
1,2
3,5
6,7
9,10
11,12
13,15
16

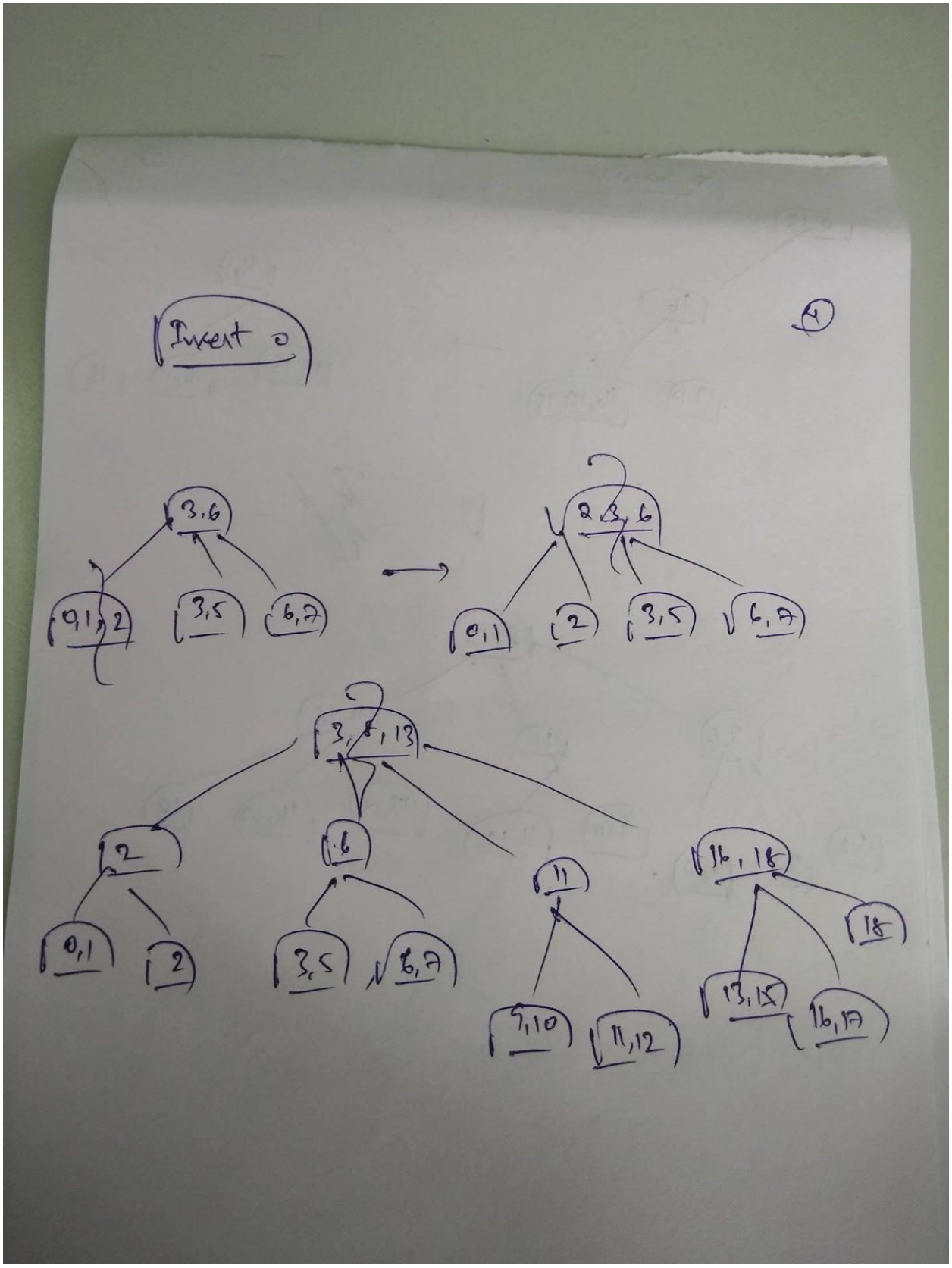
Insert 17

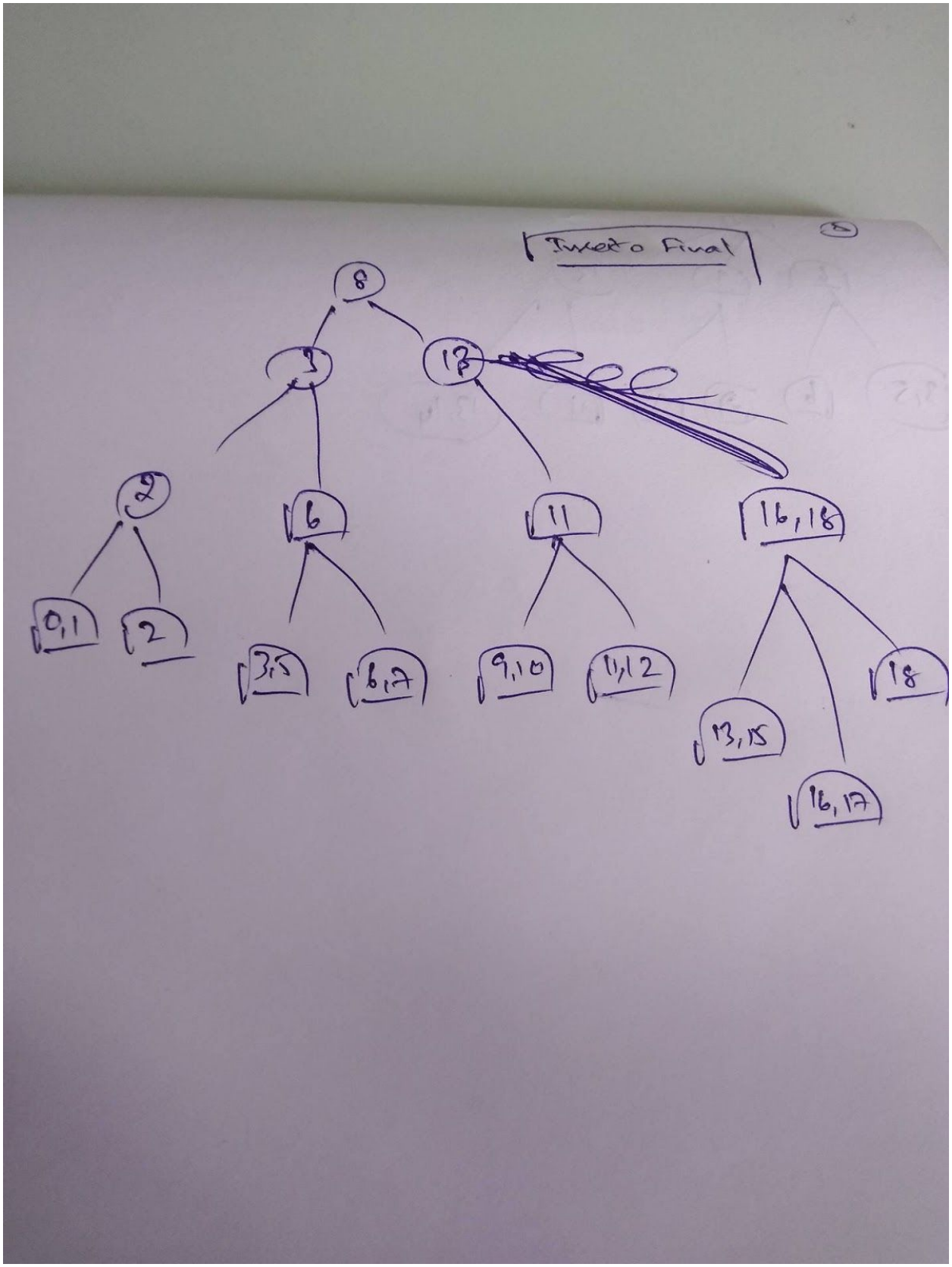


Insert 18


Insert
8,13
16
3,12
16,17,18
16,18
13,15
16,17
18
8,13
3,6
11
16,18
1,2
3,5
6,7
9,10
11,12
13,15
16,17
18

(d). Insert record with key 0 [3 Marks]




Invert o Final
8
3
13
2
6
11
16,18
0,1
2
3,5
6,7
9,10
11,12
13,15
16,17
18

(e). Delete elements (11,12,16) [Assume the deletion algorithm tries to merge/redistribute with the right sibling if one exists] [2 Marks]

Marking based on correctness and approach as well.

Question 2

a. can be 24 or 36 (if last insertion caused a split). or 17, 30, or 11 otherwise.
Explanation: should be the last element in any bucket.
Also consider correct if assumption is clearly written - that order of insertion is not necessarily the order of elements in the buckets, in which case the answer can be any element.
Also consider correct if insertion is assumed to happen at the head of the bucket, in which case the ans can be the first element of any bucket.
Answer: can be 24, 36, 17, 30 or 11.
1.5 marks for explanation
2 marks for final answer

b. Explanation: can be 24 or 36, as overflow at bucket 0 caused the split.
Also consider correct if assumption is clearly written - that order of insertion is not necessarily the order of elements in the buckets, in which case the answer can be any element out of 32, 8, 24, 44, 36.
Also consider correct if insertion is assumed to happen at the head of the bucket, in which case the ans can be 32 or 44
Answer: can be 24 or 36
1.5 marks for explanation
2 marks for final answer

c. Explanation: goes in bucket 4 as 4%8 = 4.
1.5 marks if written '4 goes to bucket 4 as 4%8 = 4'.
2 marks if written 'everything is same, only change is '4 goes to bucket 4 as 4%8 = 4'
2 marks if the index is reproduced, showing the next pointer as well, with 4 inserted in bucket 4 and 4%8 explanation is given.

d. Explanation: 15%8=7, so consider last 2 bits => goes to bucket 3 => causes overflow => split to create bucket 5(101) as next is pointing to 01. update next to point to 10 and insert 15 in overflow block at bucket 3.
1.5 marks for explanation
2 marks for final answer (including next pointer clearly shown).
Consider both correct whether 4 is inserted in bucket 4 or not.

e. Explanation: 44 and 36 are removed. everything else is same.
Consider both correct if both 4 and 15 are inserted or none of them are.
1.5 marks for explanation
2 marks for final answer (including next pointer clearly shown).

Question 3

For x - partition every 20 units
For y - partition every 50 units

a) 310<x and x<400 => need to query (400-300)/20 columns = 5 columns
520<y and y<730 => need to query (750-500)/50 rows = 5 rows
=> Total number of buckets to examine = 5*5 = 25

b) If each bucket takes n1 disk access (on an average including overflow buckets too)
=> 25*n1 disk accesses are required
If accessing the grid directory takes n2 disk accesses
=> The total number of disk access = 25*n1 + n2
(n2 can be 0, if you assume grid directory is already in main memory

Ex: marks are given for 25,25+1,25*n+1 ..

Marks are given leniently for other valid assumptions too.
(If you think marks are not given accordingly please write your explanation in query sheet during distribution)
(Marks are given for those who made trivial assumptions but didn't mention too)

Marking:

Part-a (6 marks)
* If Explanation is correct 4 marks
* If Answer is correct 2 marks

Part-b (4 marks)

* If answer is correct based on valid assumption made then 4 marks

* Partial marks will be given based on correctness of answers

(If answer is wrong in part-a but correct explanation is written in part-b then no marks will be deducted in part-b)

Question 4

There are two kinds of nested loop joins: tuple based nested loop joins and block based nested loop join. According to the prescribed textbook, block based nested loop join is simply referred to as as nested loop join since it is most commonly implemented in practise. Part (a) asks for nested loop join and part(b) asks for block nested loops, and hence according to the textbook, both of these questions ask for the same thing. However due to the subtle ambiguity involved, marks have been given if the student has written any of the two nested loop joins for part (a).

T(s): Total number of tuples in relation s
B(r): Total number of block occupied by relation s

Part (a): [3 marks]

If you consider nested loop join as ***tuple based nested loop*** join under the assumption that the relations under consideration are not clustered. Then the number of disk I/O's required are T(s) * T(r) = 2,000*20 * 5,000*5 = 40,000 * 25,000 = 1000000000

If you considered that the relations under consideration are clustered, then number of disc I/Os = B(r)*B(s) = 2000 * 5000 = 10^7

If you consider nested loop join as ***block based nested loop join*** under the assumption that the relations under consideration are clustered.
B(r) = 2000
B(s) = 5000
M = 402

We will use 401(402-1) buffers of main memory to buffer r into 401 block chunks
Outer loop iterates 2000/401 = 4.987 ~ 5 times
In the first 4 iterations , # disc I/O's = 4 * (401 + 5000) = 4 * 5401 = 21604
In the 5th iteration , #disc I/O's = (396 + 5000) = 5396
Therefore, total number of disc I/O's = 21604 + 5396 = 27,000

Part (b): [4 marks]

Same as the solution for block based nested loop join described above. Also, even if you have taken any suitable approximation, marks have been awarded.

Part(c): [8 marks]

In the question, *V(s,b): number of tuples in s having distinct values for b attribute*, is not provided. Hence you can assume this value to be a variable or some constant. I assume V(s,b) to be the variable **x.**

In case of clustered index:

#disc I/Os for join = (#disc I/Os for reading relation r) + (#disc I/Os for comparing tuples)

#disc I/Os for reading relation r = 2,000
#disc I/Os for comparing tuples = T(r)*max(1, B(s)/V(s,b)) = 2000*20*max(1, 5000/x)
= 40,000*max(1, 5000/x)
Therefore, #disc I/Os for join = 2000 + 40,000*max(1, 5000/x)

In case of non clustered index:

#disc I/Os for join = (#disc I/Os for reading relation r) + (#disc I/Os for comparing tuples)

#disc I/Os for reading relation r = 2,000
#disc I/Os for comparing tuples = T(r)*T(s)/V(s,b) = 2000*20*(5000*5)/x =1000000000/x
Therefore, #disc I/Os for join = 2000 + 1000000000/x

Question 5

TRIP(fromAddrID: INTEGER, toAddrID: INTEGER, date: DATE)
ADDRESS(id: INTEGER, street: STRING, townState: STRING)

a) SQL query to return street of all addresses in 'Stony Brook NY' that are destination of a trip on '5/14/02'
   Select street from address, trip where trip.toAddrID=address.id and address.townState='Stony Brook NY' and trip.date='05-14-2002'
   (Marks given for any equivalent query, all date formats accepted) [2]
b) Translate SQL query into corresponding 'naive' relational algebra expression
   $\pi_{street}(\sigma_{townState='Stony\ Brook\ NY'\ \wedge\ date='05\text{-}14\text{-}2002'}(trip \bowtie_{toAddrID=id} address))$
   (Or anything equivalent) [2]
c) Draw query tree for that expression
   π_{street}---$\sigma_{townState='Stony\ Brook\ NY'\ \wedge\ date='05\text{-}14\text{-}2002}$---$\bowtie_{toAddrID=id}$----trip
   \----address
   (Or anything equivalent) [2]

d) Translate expression from b into equivalent expression using pushing of selections, projections

$\pi_{street}((\pi_{id, street}(\sigma_{townState='Stony\ Brook\ NY'}, address)) \bowtie_{toAddrID=id} (\pi_{toAddrID}(\sigma_{date='05\text{-}14\text{-}2002'}, trip)))$

(For the query in b) [2]

e) Translate relational algebra expression in c into a most directly corresponding SQL query

Select street from (select id, street from address where townState='Stony Brook NY') as a, (select toAddrID from trip where date='05-14-2002') as t where a.id=t.toAddrID

(Or anything equivalent. Join..on treated equivalently to from..where)