1) [Revised Solution]

a)

i) **True.** For sequential I/O, the head points to the next block after the current block is accessed. This results in the angle to rotate to be 0 and hence the rotational delay is approximately 0.

ii) **True.** This is possible as table tuples can have variable format and made possible with the use of record headers to indicate the length and type of record.

iii) **False.** The entire block must be read even if only 1 record is required, so it will still take 1 I/O cost if only reading of 1 record is required.

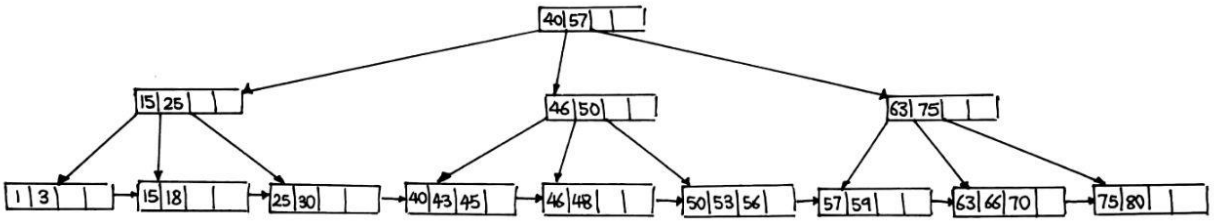
iv) **True.** There is no restriction on the number of unclustered index, so there can be more than 1 unclustered index and all unclustered indexes have to be dense index.

b) Since prefetching algorithm is used,

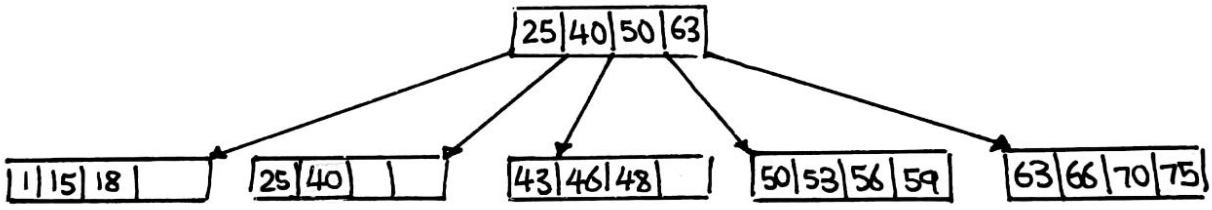
I/O cost = R + P + (# of blocks – 1) * max (R, P)
= **5R + P** [since R > P]

c)

i)



ii) Internal node: min # of pointers = 3 (min # of key = 2)
Leaf node: min # of pointers = 2 (min # of key = 2)



iii) It first finds the leaf node where the key 29 should be. It will traverse from the root node down the levels to reach that particular leaf node. Afterwards, it will search the adjacent right leaf nodes via the sequential pointers until key 55 or a key bigger than 55 is reached.

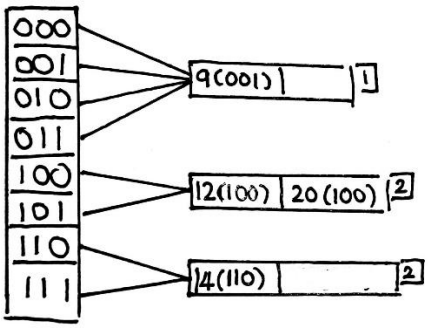
iv) The minimum disk access for figure 1 is 2 + 3 = **5**. The maximum disk access is when all the leaf nodes are within range, which is # of levels + # of leaf nodes – 1 = 3 + 6 – 1 = **8**.

2)

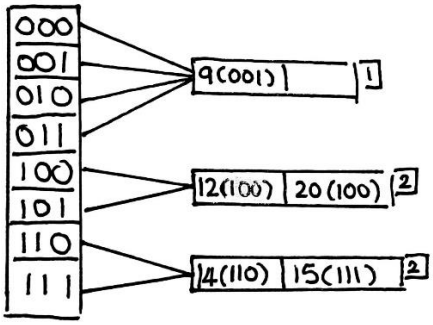
a) [Revised Solution]

i) $h(12) = 4 = 100_2$
$h(9) = 1 = 001_2$
$h(20) = 4 = 100_2$

h(14) = 6 = 110_2



ii) h(15) = 7 = 111_2



b) Number of blocks to store the pointers = 2000 / 50 = 40 buckets

c) <NO SOLUTION>

d) <NO SOLUTION>

e) 1250

3)

a)

i) I/O cost: both are 3(B(R) + B(S))

Space:

Refined sort merge join: B(R) + B(S) <= M^2

Partition hash join: min(B(R), B(S)) <= M^2

ii) Space is limited; space is sufficient; hash function is carefully designed.

b)

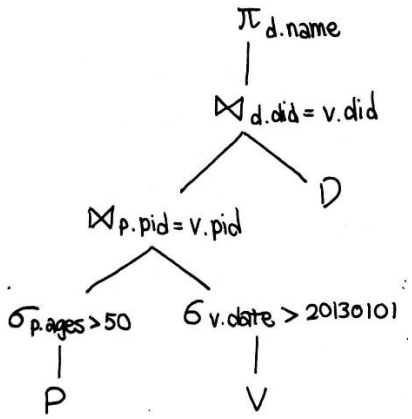
i) 570

ii) 270

c)

i) 66.6

ii)



4)

a)

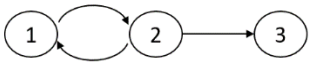
i) <START T>;<COMMIT T>;<ABORT T>;<T,X,v>.

ii) Undo logging:

The log records indicating changed database elements
The changed databases element themselves
The COMMIT log record

iii) Redo logging:

The log records indicating changed database elements
The COMMIT log record
The changed database elements themselves

b) 

(There is cycle between 1 and 2. Thus, not conflict serializable)

c)

i) Wait-die scheme:

A transaction T is waiting for a lock that is held by transaction U:
If T is older than U, TS(T) < TS(U), then T is allowed to wait;
If T is newer than U, then T "dies"; it is rolled back.

2 and 4 will die.

ii) Wound-wait scheme:

A transaction T is waiting for a lock that is held by transaction U:
If T is older than U, TS(T) < TS(U), then it "wounds" U;
If T is newer than U, then T waits for the locks held by U.

3 will be wounded

--End of Answers—

Solver: Chen Mengxuan