Student number:	2467273
Course title:	COMPSCI4013 Database Systems (H)
Questions answered:	ALL

A.

1.

a)

```
SELECT MID, MAX(Price)
FROM VEHICLE
GROUP BY MID;
```

b)

```
SELECT MID, Price
FROM VEHICLE AS I
WHERE Price >= ALL (
   SELECT Price
   FROM VEHICLE AS J
   WHERE I.MID = J.MID
);
```

c)

```
SELECT MID, COUNT(*)
FROM VEHICLE
WHERE Price > 100,000 AND MID IN (
   SELECT I.MID
   FROM VEHICLE I
   GROUP BY I.MID
   HAVING COUNT(*) > 1000)
GROUP BY MID;
```

B.

2.

a)

If n = number of clusters; and b = numbers of blocks:

- bfr = floor(B/R) = floor(512/250) = 2 records/block
- b = ceil(r/bfr) = ceil(10,000 / 2) = 5000
- n = 100
- Linear search cost for equality selection = b * (n+1) / 2n = 5000 * 101 / 200 = **2525** block accesses
- s = r/n = 10,000 / 100 = 100 employees per department
- d = 29 – 10 + 1 = 20 departments being searched
- Linear search for query = 2525 + d * s / bfr = 2525 + 20 * 100 / 2 = **3525** block accesses

b) ~~If m = bfr = 2 records/block, t = log_m(b) = ceil(log_2(5000)) = 13 levels~~

~~Cost = t + 1 + d * s / bfr = 14 + 1000 = 1014 block accesses~~

Index entry: V_DNO + P = 50 + 10 = 60 bytes

ibfr = m = floor(B/(V+P)) = floor(512/60) = 8 index entries/block

ib = ceil(r/ibfr) = ceil(10,000/8) = 1250 index blocks

b2 = ceil(ib/m) = ceil(1250/8) = 157 blocks

b3 = ceil(b2/m) = ceil(157/8) = 20 blocks

b4 = ceil(b3/m) = ceil(20/8) = 3 blocks

b5 = ceil(b4/m) = ceil(3/8) = 1 block

=> t = 5

5-level clustering index (clustering as L-1) cost = t + d / bfr = 5 + 20/2 = 15 block accesses

c) ~~m < 2^(b(n-1)/2n)~~

~~Linear search cost for equality selection = b * (n+1) / 2n~~

An equality selection from the multilevel clustering index would only require t + 1 (6) block accesses. Since the blocking factor is 2, there are 10 records in the first 5 blocks, which means that the largest record in those 10 (DNO=x=10) would be the maximum DNO value before using Clustering index would be more beneficial.

C.

3.

a)

Clustering Index over AGE in EMPLOYEE:

- Index entry: {AGE, pointer}
- Index entry size = 50 (EMPLOYEE attribute size) + V = 50 + 50 = 100 bytes
- ibfr = floor(B/50+V) = floor(256/100) = 2 entries/block
- ~~Index entries = 65 – 25 + 1 (inclusive) = 41~~
- ~~Index blocks ib = ceil(41/2) = 21~~
- Index entries = NDV(AGE) = 100
- Index entries = ceil(100/2) = 50

Clustering Index over ESSN in DEPENDENT:

- Index entry: {ESSN, pointer}
- Index entry size = 50 (DEPENDENT attributes size) + V = 100 bytes
- ibfr = floor(B/50+V) = floor(256/100) = 2
- Index entries = NDV(ESSN) = 50
- Index blocks ib = ceil(50/2) = 25

b)

Plan 1: First do range selection (AGE >= 45), then join:

- sl(AGE>=45) = (65-45)/(65-25) = 20/40 = ½
- ~~Cost(AGE) = t + 1 + r*sl(AGE) = 1 + 1 + 1000 * ½ = 502 block accesses < 5000 (memory)~~
- Cost(AGE) = log2(ib_AGE) + ceil(r*sl(AGE)/f) – 1 = log2(50) + ceil(1000 * ½ / 2) – 1 = 5.64 + 250 – 1= 254.64 block accesses < 5000 (memory)

Plan 2: First join relations, then select:

- Join with index-based Nested Loop Join (choose DEPENDENT as outer relation since r_D < r_E)
- n_B = 5000 (memory) blocks
- Cost(JOIN) = r_D + r_E * ceil(r_D/(n_B - 2)) = 500 + 1000 * ceil(500/4998) = 1500 block accesses

In conclusion, Plan 1 (first select, then join) is better since 254.64 < 1500