Quiz 5

Deadline	Friday, 24 April 2020 at 11:59PM
Latest Submission	Friday, 17 April 2020 at 5:46PM
Maximum Mark	4

Question 1 (1 mark)

Consider a table of machine parts (*Parts(id,name,colour,size,cost)*), about which we have the following information:

```
db=# select count(distinct size) from Parts;
 count
-------
    5

db=# select min(size) from Parts;
  min
------
 tiny

db=# select max(size) from Parts;
  max
------
 huge

db=# select count(*) from Parts;
 count
-------
  6507
```

Approximately how many tuples would you expect to find in the result of the following query?

```
select * from Parts where size = 'huge';
```

Assume a uniform distribution of part sizes.

(a) ◯	650
(b) ◯	3253
(c) ◯	6506
(d) ◯	None of the other options is correct.

(e) ◉	1301

Question 2 (1 mark)

Consider the three relations *R(id,a,b)*, *S(rid,tid,c)*, *T(id,d,e,f)* where the *id* attributes are primary keys, and *S.rid* is a foreign key referencing *R.id* and *S.tid* is a foreign key referencing *T.id*.

Consider also the following join on these three relations:

```
select a,b,c,d,e,f from R, S, T where R.id = S.rid and T.id = S.tid
```

Which of the following does **not** represent a possible join order for the above SQL statement?

(a) ○	(R ⋈ S) ⋈ T
(b) ○	R ⋈ (S ⋈ T)
(c) ○	(S ⋈ T) ⋈ R
(d) ◉	S ⋈ (T ⋈ R)
(e) ○	They are all valid join orders for the query.

Question 3 (1 mark)

Consider the following two relations/tables:

R(a,b,c) where r_R = 100000, c_R = 100, b_R = 1000

S(c,d,e) where r_S = 50000, c_S = 500, b_S = 100

If we do a natural join on these two tables, using a block nested loop join with 35 buffers, how many pages do we *read* in completing the join?

(a) ○	1100
(b) ○	100000
(c) ○	None of the other options is correct
(d) ○	3100
(e) ◉	4100

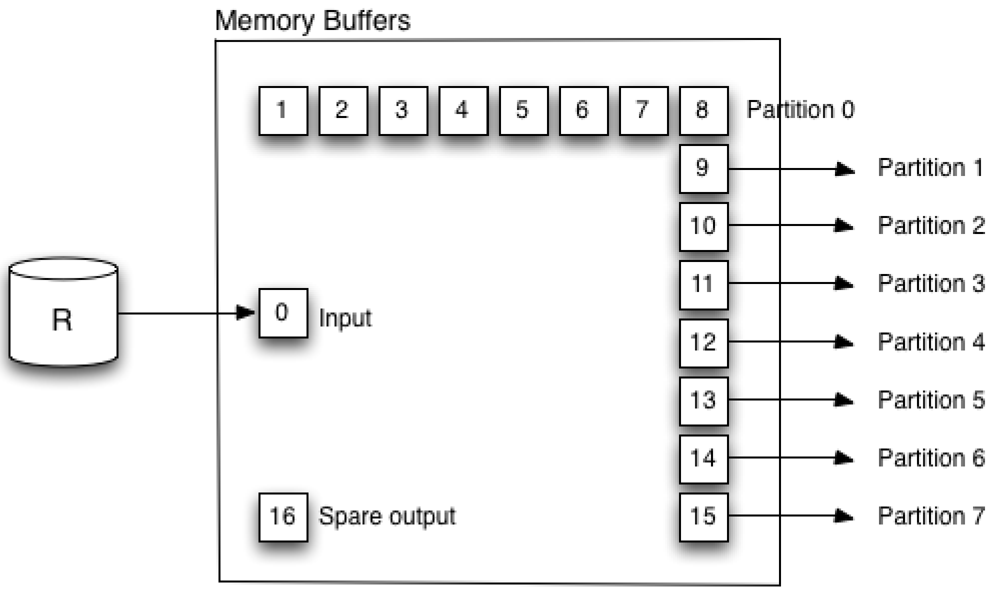
Question 4 (1 mark)

Consider two tables *R(id,x,y,z)* and *S(id,a,b,rid)* where *rid* is a foreign key referring to *R.id*. Consider also the following join on these tables:

```
select x,a from R, S where R.id = S.rid;
```

This join is implemented using the hybrid hash join algorithm with 17 memory buffers, 1 buffer used for input (#0), 8 buffers used to hold one partition in memory (#1-#8), 7 output buffers used to transfer tuples in the other partitions to disk (#9-#16), and 1 "spare" output buffer (#16) used only in the first phase (see below).

The diagram below shows the first phase of the hash join, where *R* is scanned and partitioned into 8 hash buckets. Partition 0 is stored in memory; partitions 1-7 are written to disk.



After the first phase has partitioned *R*, the join algorithm then partitions *S* using the same hash function. Any tuples that hash to partition 0 are matched against *R*'s partition 0 tuples held in memory buffers 1-8, and any resulting matches are written to disk via the spare output buffer. Other *S* tuples are written to disk-based partitions. Partitions 1-7 for *R* and *S* are then processed as for a standard hash join, using a second hash function.

Assume that:

- we have a well-behaved (uniform) hash functions for both the first and second phases
- *R* contains 3000 tuples in 60 pages; each partition of *R* requires exactly 8 pages
- *S* contains 1600 tuples in 40 pages; each partition of *S* requires exactly 5 pages
- each tuple in *S* refers to a different tuple in *R* (i.e. *S.rid* is unique)
- the result contains 1600 tuples which are written into 30 pages (i.e. they count as disk I/O)

Based on the above, compute the number of disk I/Os needed to execute this join.

(a)	308
(b)	272 k partitions on disk 60 + 40 + 2 * K * (ceil(br/ total partitions) + ceil(bs/ total partitions)) = 60 + 40 + 2 * 7 * (8 + 5) = 282
(c)	312 plus final write cost + 30 = 312 60 + 40 + 7 * 8 writes + 7 * 5 reads + 7 * 8 reads + 7 * 5 writes + 30 writes to disk
(d)	338
(e)	None of the other options is correct

Submit