

Q1.

```
$ ccm create -n 5 single_dc
```

```
$ ccm node1 ring
```

```
Datcenter: datacenter1
=====
```

Address	Rack	Status	State	Load	Owns	Token	initial_token
127.0.0.1	rack1	Up	Normal	97.96 KiB	40.00%	-9223372036854775808	
127.0.0.2	rack1	Up	Normal	136.71 KiB	40.00%	-5534023222112865485	
127.0.0.3	rack1	Up	Normal	130.68 KiB	40.00%	-1844674407370955162	
127.0.0.4	rack1	Up	Normal	135.61 KiB	40.00%	1844674407370955161	
127.0.0.5	rack1	Up	Normal	97.91 KiB	40.00%	5534023222112865484	

Figure1: The output of the "ccm node1 ring" command

Q2.

a) **endpoint_snitch**: SimpleSnitch

b) **initial_token**: -9223372036854775808

The **initial_token** property value is shown in the first line's "**Token**" column of **Figure 1** above by executing "ccm node1 ring" command.

c) **partitioner**: org.apache.cassandra.dht.Murmur3Partitioner.

d) **rpc_address**: 127.0.0.1

The **rpc_address** property value is shown in the first line's "**Address**" column of **Figure 1** above by executing "ccm node1 ring" command.

Q3.

Below is cassandra-topology.properties file's content.

```
# Cassandra Node IP=Data Center:Rack
192.168.1.100=DC1:RAC1
192.168.2.200=DC2:RAC2

10.0.0.10=DC1:RAC1
10.0.0.11=DC1:RAC1
10.0.0.12=DC1:RAC2

10.20.114.10=DC2:RAC1
10.20.114.11=DC2:RAC1

10.21.119.13=DC3:RAC1
10.21.119.10=DC3:RAC1

10.0.0.13=DC1:RAC2
10.21.119.14=DC3:RAC2
10.20.114.15=DC2:RAC2

# default for unknown nodes
default=DC1:r1

# Native IPv6 is supported, however you must escape the colon in the IPv6 Address
# Also be sure to comment out JVM_OPTS="$JVM_OPTS -Djava.net.preferIPv4Stack=true"
# in cassandra-env.sh
fe80\:\0\:\0\:\0\:202\:\b3ff\:\fe1e\:\8329=DC1:RAC3
```

There is no obvious relationship between the content of the file and the output of the "ccm node1 ring" command.

Q4.

a) Create a keyspace

```
CREATE KEYSPACE IF NOT EXISTS ass2 WITH REPLICATION={ 'class': 'SimpleStrategy',  
'replication_factor':3};
```

b)

```
$ SOURCE 'table_declarations.cql';
```

```
COPY driver(driver_name,current_position,email,mobile,password,skill) FROM  
'~/private/a2/driver_data.txt' ;
```

```
COPY vehicle(vehicle_id,status,type) FROM '~/private/a2/vehicle_data.txt' ;
```

```
COPY time_table(line_name,service_no,time,distance,latitude,longitude,stop) FROM  
'~/private/a2/time_table_data.txt' ;
```

```
COPY data_point(line_name,service_no,date,sequence,latitude,longitude,speed) FROM  
'~/private/a2/data_point_data.txt' ;
```

```
$ DESCRIBE tables
```

```
time_table data_point driver vehicle
```

```
$ select * from driver where driver_name='pavle';
```

driver_name	current_position	email	mobile	password	skill
pavle	Upper Hutt	pmogin@ecs.vuw.ac.nz	213344	pm33	{'Ganz Mavag', 'Guliver', 'Matangi'}

```
$ select * from vehicle where vehicle_id='FA1122';
```

vehicle_id	status	type
FA1122	Upper Hutt	Ganz Mavag

```
$ select * from data_point where line_name='Hutt Valey Line' and service_no=2 and  
date=20160322;
```

line_name	service_no	date	sequence	latitude	longitude	speed
Hutt Valey Line	2	20160322	2016-03-21 21:37:50.000000+0000	-41.2272	174.77	29.1


```
$ select * from time_table where line_name='Hutt Valley Line' and service_no=2 and time=1045;
```

line_name	service_no	time	distance	latitude	longitude	stop
Hutt Valley Line	2	1045	34.3	-41.2865	174.7762	Wellington

Q5.

a) [~]% ccm node1 nodetool getendpoints ass2 driver pavle

```
127.0.0.1
127.0.0.2
127.0.0.3
```

b)

```
[~]% ccm node4 cqlsh
```

```
$ CONSISTENCY ALL
```

```
$ select * from driver where driver_name='pavle';
```

```
-----+-----+-----+-----+-----+-----
(1 rows)
```

```
[~]% ccm node1 stop
```

```
[~]% ccm node4 cqlsh
```

```
$ select * from driver where driver_name='pavle';
```

What I have learned is when the consistency level is ALL, the read operation will fail if one of replica does not respond.

c)

```
$ CONSISTENCY QOURUM
```

```
$ select * from driver where driver_name='pavle';
```

```
-----+-----+-----+-----+-----+-----
(1 rows)
```

```
[~]% ccm node2 stop
```

```
[~]% ccm node4 cqlsh
```

```
$ select * from driver where driver_name='pavle';
```

```
-----+-----+-----+-----+-----+-----
(1 rows)
```

What I have learned is when the consistency level is QOURUM, the read operation will success if the number of replicas gives respond $\geq q$, whereas the read operation will fail if the number of replicas gives respond $< q$. (In this case, $q=[3/2]+1=2$).

d)

```
$ CONSISTENCY One
```

```
$ select * from driver where driver_name='pavle';
```

```
-----+-----+-----+-----+-----+-----
(1 rows)
```

```
[~]% ccm node3 stop
```

```
[~]% ccm node4 cqlsh
```

```
$ select * from driver where driver_name='pavle';
```

```
-----+-----+-----+-----+-----+-----
(1 rows)
```

What I have learned is when the consistency level is ONE, the read operation will success if any replica gives respond.

Q6.

1) Using partition key (driver_name) to get the data's token.

```
$ SELECT token(driver_name), driver_name FROM driver where driver_name='pavle';
```

```
system.token(driver_name) | driver_name
-----+-----
8952481895309574844 |    pavle
```

2) Run "ccm node1 ring" command

```
Datacenter: datacenter1
=====
Address  Rack  Status State  Load      Owns           Token
-----
127.0.0.1 rack1  Up    Normal  97.96 KiB  40.00%        5534023222112865484
127.0.0.2 rack1  Up    Normal  136.71 KiB 40.00%        -9223372036854775808
127.0.0.3 rack1  Up    Normal  130.68 KiB 40.00%        -5534023222112865485
127.0.0.4 rack1  Up    Normal  135.61 KiB 40.00%        -1844674407370955162
127.0.0.5 rack1  Up    Normal  97.91 KiB  40.00%        1844674407370955161
127.0.0.5 rack1  Up    Normal  97.91 KiB  40.00%        5534023222112865484
```

3) From the above data, it is known that the data's token(8952481895309574844) is in the range of node1(5534023222112865484~9223372036854775807), in addition, each datacenter has a replication factor of 3, so the nodes that store replicas of the data(the driver pavle) should be node1, node2 and node3.

Q7.

This is named the hinted handoff which is a Cassandra mechanism that writes a hint and data to the coordinator node if some of n replicas are down or not replying.

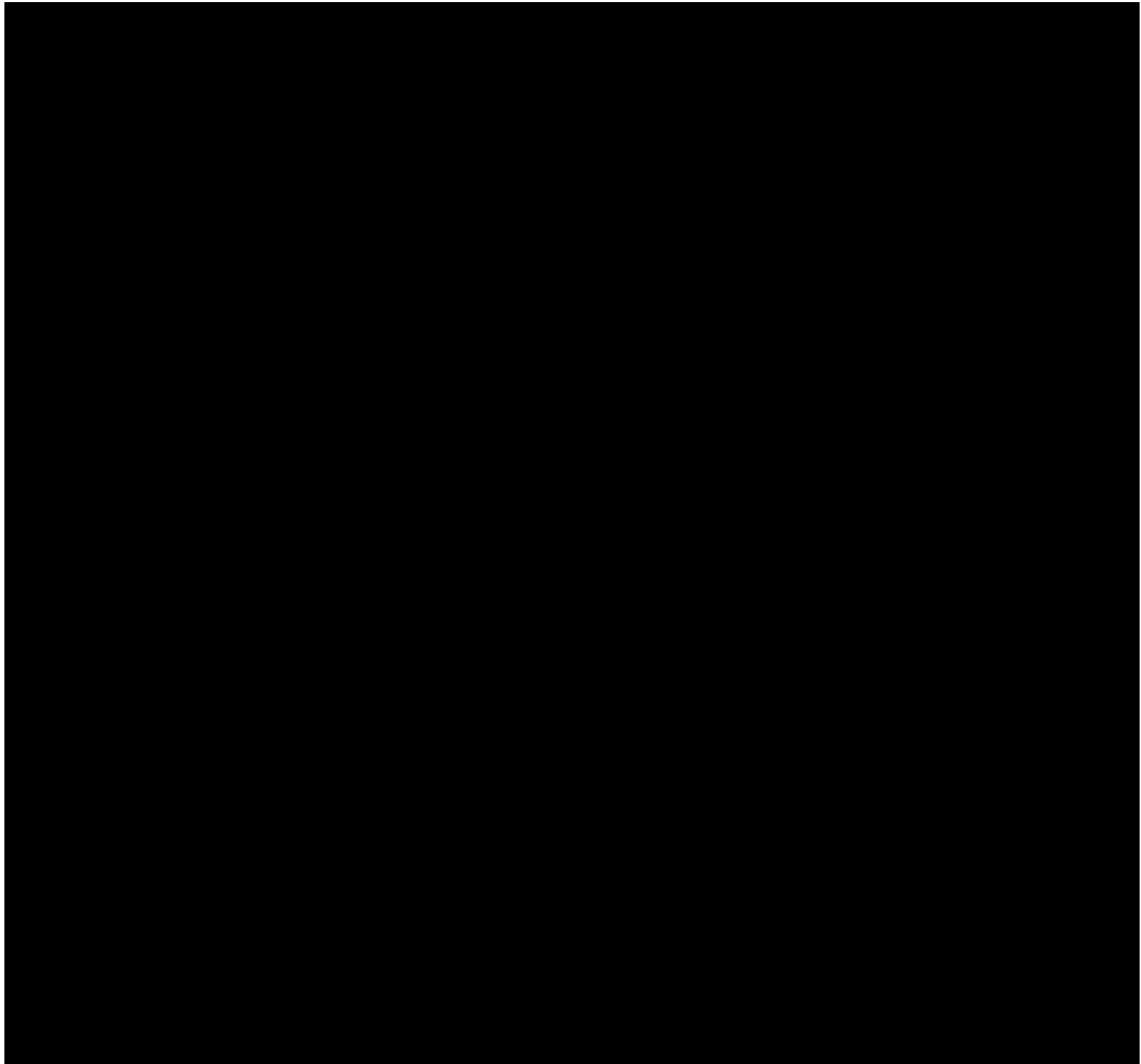
The brief explanation as follows,

1) During a write operation, because node4 was down, the coordinator node(node3) stored a hint and james's data in the local system.hints table, also wrote james's data to node1 and node5.

2) After node5 and node1 went down and the node4 started, the coordinator node(node3) sent the hinted data (james's data) to the node4 and the data (james's data) was written to the node4.

3) So when a client(c1) via node3 and sent the read statement, because the latest james's data has been written to the node4, the client can get the latest james's data successfully.

Q9.



Q10.

endpoint_snitch: org.apache.cassandra.locator.PropertyFileSnitch

SimpleSnitch is used for single-data centre deployments.

PropertyFileSnitch is used for multiple-data centre deployments.

Q11.

Below is cassandra-topology.properties file's content.

```
default=dc1:r1
127.0.0.1=dc1:r1
127.0.0.2=dc1:r1
127.0.0.3=dc1:r1
127.0.0.4=dc1:r1
127.0.0.5=dc1:r1
127.0.0.6=dc2:r1
127.0.0.7=dc2:r1
127.0.0.8=dc2:r1
127.0.0.9=dc2:r1
```

Each node's address, rack and datacenter information are the same between the content of "cassandra-topology.properties" file and the output of the "ccm node1 ring" command.

Q12.

```
CREATE KEYSPACE IF NOT EXISTS ass2 WITH
REPLICATION={'class':'NetworkTopologyStrategy', 'dc1':3, 'dc2':3};
```

Q13.

\$ DESC tables

```
time_table driver
```

\$ select * from driver where driver_name='pavle';

```
driver_name | current_position | email          | mobile | password | skill
-----+-----+-----+-----+-----+-----
    pavle |   Upper Hutt | pmogin@ecs.vuw.ac.nz | 213344 |   pm33 | {'Ganz Mavag', 'Guliver',
'Matangi'}
```

\$select * from time_table where line_name='Hutt Valley Line' and service_no=2 and time=1045;

```
line_name    | service_no | time | distance | latitude | longitude | stop
-----+-----+-----+-----+-----+-----+-----
Hutt Valley Line |      2 | 1045 |   34.3 | -41.2865 | 174.7762 | Wellington
```

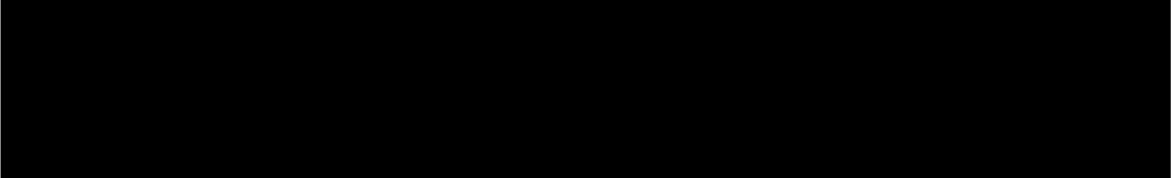
Q14.

i.

```
[~] % ccm node1 cqlsh
```

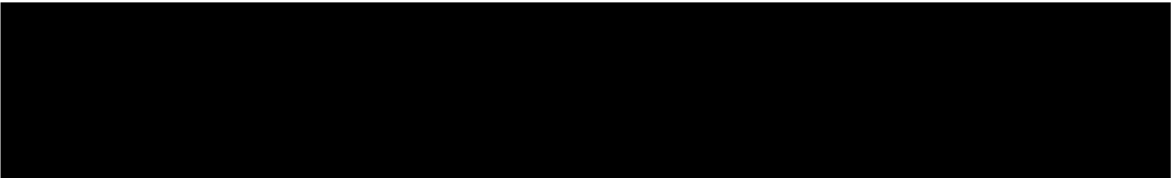
```
$ CONSISTENCY QUORUM
```

```
$ select driver_name,password from driver where driver_name='pavle';
```



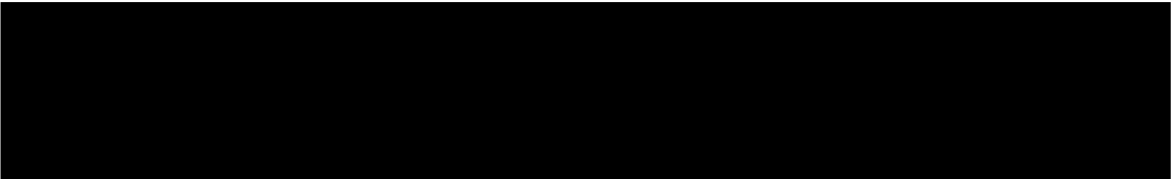
```
$ CONSISTENCY EACH_QUORUM;
```

```
$ select driver_name,password from driver where driver_name='pavle';
```



```
$ CONSISTENCY LOCAL_QUORUM;
```

```
$ select driver_name,password from driver where driver_name='pavle';
```



```
$ exit;
```

```
[~] % ccm node6 cqlsh
```

```
$ CONSISTENCY LOCAL_QUORUM;
```

```
$ select driver_name,password from driver where driver_name='pavle';
```



ii.

Note: The driver pavle's data are stored in node1,node2,node3,node6,node7,node8.


```
[~]% ccm node7 stop
```

```
[~]% ccm node8 stop
```

```
[~]% ccm node6 cqlsh
```


```
$ CONSISTENCY QUORUM
```

```
$ select driver_name,password from driver where driver_name='pavle';
```




```
$ CONSISTENCY EACH_QUORUM;
```

```
$ select driver_name,password from driver where driver_name='pavle';
```



```
$ CONSISTENCY LOCAL_QUORUM;
```

```
$ select driver_name,password from driver where driver_name='pavle';
```



```
$ exit;
```

```
[~] % ccm node1 cqlsh
```

```
$ CONSISTENCY LOCAL_QUORUM;
```

```
$ select driver_name,password from driver where driver_name='pavle';
```



From above experiments, what I have learned is the following:

1) When consistency level is QURUM, the read operation will success if the number of replicas gives respond $\geq q$ (in this case, $q=[6/2]+1=4$) regardless of data centre.

2) When consistency level is consistency EACH_QURUM, the read operation will fail if the number of replicas gives respond $< q$ (in this case, $q=\lceil 3/2 \rceil + 1 = 2$) in each data centre.

3) When consistency level is consistency LOCAL_QURUM, the read operation will fail if the number of replicas gives respond $< q$ (in this case, $q=\lceil 3/2 \rceil + 1 = 2$) in the current data centre.

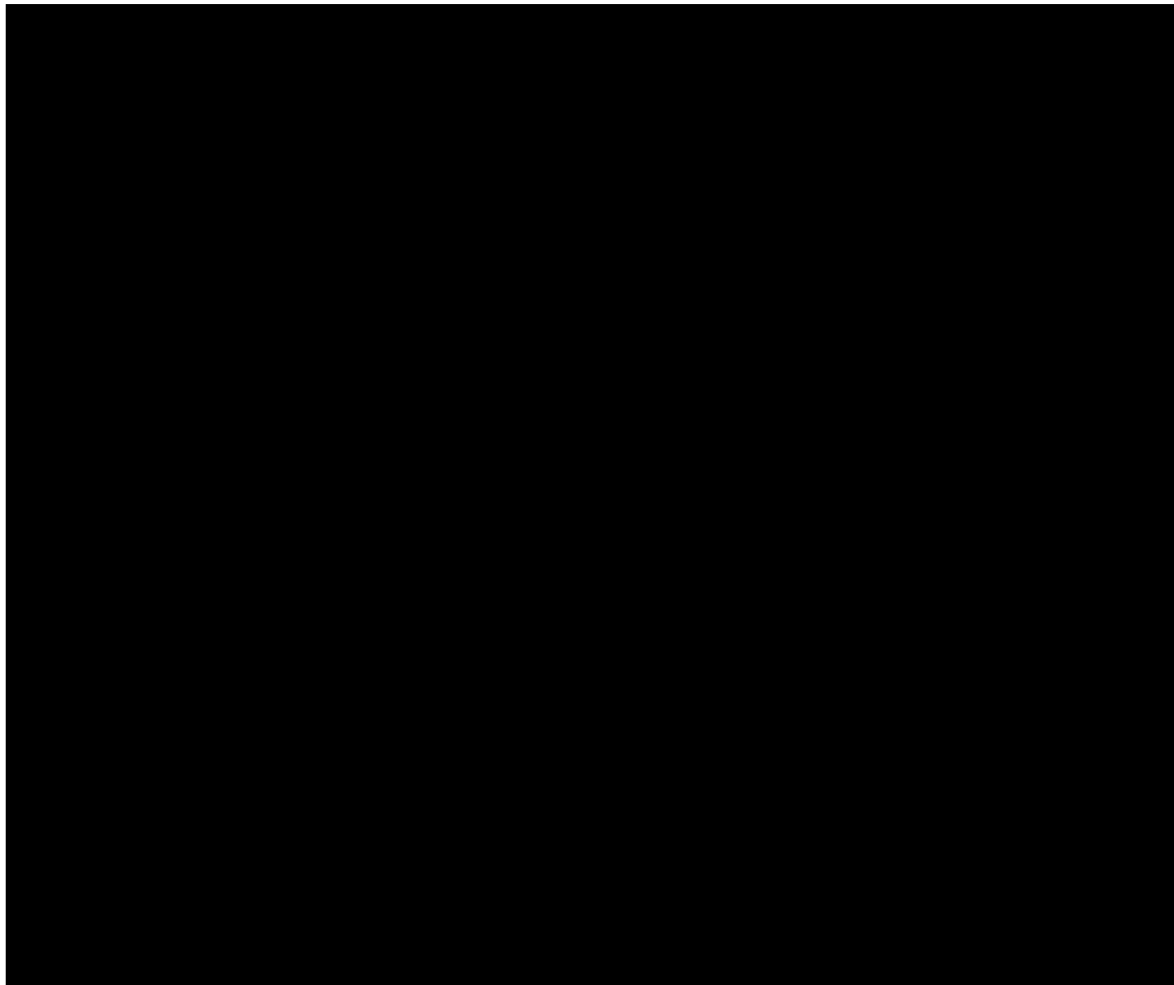
Q15.

1) Using partition key (line_name and service_no) to get the data's token.

```
$ SELECT token(line_name,service_no), line_name, service_no FROM time_table;
```

```
system.token(line_name, service_no) | line_name      | service_no
-----+-----+-----
2322329569350831795 | Hutt Valley Line |      2
```

2) Run "ccm node1 ring" command.



3) From the above data, it is known that the data's token(2322329569350831795) is in the range of node5(1844674407370955161~5534023222112865483) and node9(100~4611686018427388003), in addition, each datacenter has a replication factor of 3, so the nodes that store replicas of the data(line_name='Hutt Valley Line ' and service_no=2) should be node5, node1, node2, node9, node6, node7.