Test Output File

*Descriptions about testcases are shown in '/test/README.py'

*All messages that are printed using '[' and ']' follows the following format:
['Class name of stub instance'] 'request name'('parameters'): 'request result'

- If a class name ends with 'Stub', then the instance is the client side of the RPC call.
- If a class name ends with 'Servicer', then the instance is the server side of the RPC call.

- catalog component output

```
(cs677) imjeonghun@macmini catalog % python3 catalog.py
[CatalogServicer] Query(Tux): {'price': '14.99', 'quantity': 3}
(Buy Successful) Tux: (before: 3) -> (after: 2)
[CatalogServicer] Order(Tux, 1): {'order_result': 1}
[FrontStub] Invalidate(Tux)
[CatalogServicer] Query(Tux): {'price': '14.99', 'quantity': 2}
(Buy Successful) Tux: (before: 2) -> (after: 1)
[CatalogServicer] Order(Tux, 1): {'order_result': 1}
[FrontStub] Invalidate(Tux)
(Buy Successful) Tux: (before: 1) -> (after: 0)
[CatalogServicer] Order(Tux, 1): {'order_result': 1}
[FrontStub] Invalidate(Tux)
Restocking ['Tux', '14.99', 0] -> ['Tux', '14.99', 100]
[FrontStub] Invalidate(Tux)
[CatalogServicer] Query(Tux): {'price': '14.99', 'quantity': 100}
```

-order component 1 output

```
(cs677) imjeonghun@macmini order % COMPONENT_ID=1 ORDER_LOG_FILE=data/log1.csv python3 order.py
[RecoveryServicer] BackOnline(0): 3
[RecoveryServicer] BackOnline(0): 3
[RecoveryServicer] RequestMissingLogs(1): (1, Whale, 5)
[RecoveryServicer] RequestMissingLogs(2): (2, Tux, 1)
[OrderSerivcer] Leader selected. (LEADER_ID: 1, IS_LEADER: True)
[CatalogStub] Order(Tux, 1): {'order_result': 1}
[OrderSerivcer] Buy(Tux, 1): {'order_number': 3}
[OrderSerivcer] Check(3): {'product_name': 'Tux', 'quantity': 1}
Turning inactive...
```

(After restarting the component)

```
(cs677) imjeonghun@macmini order % COMPONENT_ID=1 ORDER_LOG_FILE=data/log1.csv python3 order.py
############################## TEST ##############################
[Testcase 11] Filling missing logs.
[RecoveryStub 2] BackOnline(0): 6
[RecoveryStub 2] RequestMissingLogs(4): (4, Tux, 1)
[RecoveryStub 2] RequestMissingLogs(5): (5, Tux, 1)
[RecoveryStub 3] BackOnline(0): 6
##################################################################
```

-order component 2 output

```
(cs677) imjeonghun@macmini order % COMPONENT_ID=2 ORDER_LOG_FILE=data/log2.csv python3 order.py
[RecoveryStub 1] BackOnline(0): 3
[RecoveryServicer] BackOnline(0): 3
[OrderSerivcer] Leader selected. (LEADER_ID: 1, IS_LEADER: False)
[OrderSerivcer] Propagate(3, Tux, 1): {'ping_number': 0}
[OrderSerivcer] Leader selected. (LEADER_ID: 2, IS_LEADER: True)
[CatalogStub] Order(Tux, 1): {'order_result': 1}
[OrderSerivcer] Buy(Tux, 1): {'order_number': 4}
Propagate to component 1 failed
[CatalogStub] Order(Tux, 1): {'order_result': 1}
[OrderSerivcer] Buy(Tux, 1): {'order_number': 5}
Propagate to component 1 failed
[RecoveryServicer] BackOnline(0): 6
[RecoveryServicer] RequestMissingLogs(4): (4, Tux, 1)
[RecoveryServicer] RequestMissingLogs(5): (5, Tux, 1)
```

-order component 3 output

```
(cs677) imjeonghun@macmini order % COMPONENT_ID=3 ORDER_LOG_FILE=data/log3.csv python3 order.py
############################## TEST ##############################
[Testcase 1] Filling missing logs.
[RecoveryStub 1] BackOnline(0): 3
[RecoveryStub 1] RequestMissingLogs(1): (1, Whale, 5)
[RecoveryStub 1] RequestMissingLogs(2): (2, Tux, 1)
[RecoveryStub 2] BackOnline(0): 3
#################################################################
[OrderSerivcer] Leader selected. (LEADER_ID: 1, IS_LEADER: False)
[OrderSerivcer] Propagate(3, Tux, 1): {'ping_number': 0}
[OrderSerivcer] Leader selected. (LEADER_ID: 2, IS_LEADER: False)
[OrderSerivcer] Propagate(4, Tux, 1): {'ping_number': 0}
[OrderSerivcer] Propagate(5, Tux, 1): {'ping_number': 0}
[RecoveryServicer] BackOnline(0): 6
```

-front-end output

```
(cs677) imjeonghun@macmini front-end % python3 front_end.py
############################## TEST ##############################
[Testcase 2] Leader Selection
Choosing the leader of OrderStub...
        Leader Selected: 1
#################################################################
[CatalogStub] Query(Tux): {'price': 14.99, 'quantity': 3)
[Cache] query request(Tux): {price: 14.99, quantity: 3}
[OrderStub 1] Buy(Tux, 1): {'order_number': 3}
[FrontServicer] Invalidate(Tux): {'response': 0}
[Cache] pop(Tux)
[CatalogStub] Query(Tux): {'price': 14.99, 'quantity': 2)
[OrderStub 1] Check(Tux, 1): {'order_number': 3}
Choosing the leader of OrderStub...
Choosing the leader of OrderStub...
        Ping to Order stub 1 failed
        Leader Selected: 2
[FrontServicer] Invalidate(Tux): {'response': 0}
[Cache] pop(Tux)
[OrderStub 2] Buy(Tux, 1): {'order_number': 4}
[FrontServicer] Invalidate(Tux): {'response': 0}
[OrderStub 2] Buy(Tux, 1): {'order_number': 5}
[FrontServicer] Invalidate(Tux): {'response': 0}
[CatalogStub] Query(Tux): {'price': 14.99, 'quantity': 100)
```

-client output

```
(cs677) imjeonghun@macmini client % python3 client.py
############################## TEST ##############################
[Testcase 3] A query about the product "Tux" is handled using the catalog component.
query(Tux): [200] {'data': {'name': 'Tux', 'price': '14.99', 'quantity': 3}}

[Testcase 4] A query about the product "Tux" is handled using cache in the front-end component.
query(Tux): [200] {'data': {'name': 'Tux', 'price': '14.99', 'quantity': 3}}

[Testcase 5] A buy request for one "Tux" is successful. The information about "Tux" in front-end is invalidated.
buy(Tux, 1): [200] {'data': {'order_number': 3}}

[Testcase 6] A query about the product "Tux" is handled using the catalog component due to invalidate RPC call.
query(Tux): [200] {'data': {'name': 'Tux', 'price': '14.99', 'quantity': 2}}

[Testcase 7] A query about the previously made order is successful.
check(3): [200] {'data': {'number': '3', 'name': 'Tux', 'quantity': 1}}

[Testcase 8] During this buy request, the leader order component becomes inactive and a new leader selection is held. T
he purchase request is successfully handled by the new leader order component.
buy(Tux, 1): [200] {'data': {'order_number': 4}}

[Testcase 9] This request will make the quantity of "Tux" in catalog to be 0. Within one second the product "Tux" will
be restocked. (It is set to one second only in the test.)
buy(Tux, 1): [200] {'data': {'order_number': 5}}

[Testcase 10] After the restock has happened, the quantity of "Tux" will be 100.
query(Tux): [200] {'data': {'name': 'Tux', 'price': '14.99', 'quantity': 100}}
#################################################################
```