Task 3 (6 marks)
Processing transactions at `READ COMMITTED` level by a snapshot isolation scheduler

Consider a stored PL/SQL function `MAX_MIN` given below.

```
CREATE OR REPLACE FUNCTION MAX_MIN ( orderkey IN NUMBER )
RETURN NUMBER IS
 max_value NUMBER(12);
 min_value NUMBER(12);

BEGIN
 SELECT MAX(L_QUANTITY * L_EXTENDEDPRICE)
 INTO max_value
 FROM LINEITEM
 WHERE L_ORDERKEY = orderkey;

 SELECT MIN(L_QUANTITY * L_EXTENDEDPRICE)
 INTO min_value
 FROM LINEITEM
 WHERE L_ORDERKEY = orderkey;

 RETURN max_value-min_value;
END MAX_MIN;
/
```

(1) (4 marks)
Show a sample concurrent execution of the function at `READ COMMITTED` level that interleaves the operations with another transaction and such that a result returned by the function is incorrect. The operations performed by another transaction are up to you.

When visualizing the concurrent executions use a technique of two-dimensional diagrams presented to you during the lecture classes, for example, see a presentation 14 Transaction Processing in Oracle DBMS slide 16.

```
MAX-MIN transaction                Other transaction
EXECUTE MAX_MIN ( 1 )

SELECT MAX(L_QUANTITY * L_EXTENDEDPRICE)
INTO max_value
FROM LINEITEM
WHERE L_ORDERKEY = 1;
                                        UPDATE LINEITEM
                                        SET L_QUANTITY =
                                               (SELECT MAX(L_QUANTITY)+1
                                                FROM LINEITEM),
                                            L_EXTENDEDPRICE =
                                               (SELECT MAX(L_EXTENDEDPRICE)+1
                                                FROM LINEIITEM;
                                        WHERE L_ORDEKEY = 1;
                                        COMMIT
SELECT MIN(L_QUANTITY * L_EXTENDEDPRICE)
INTO min_value
FROM LINEITEM
WHERE L_ORDERKEY = 1;

RETURN max_value-min_value;
```

After an update performed by the other transaction a value of `MIN(L_QUANTITY * L_EXTENDEPRICE)` becomes larger than the original value of `MAX(L_QUANTITY * L_EXTENDEDPRICE)` because the other transaction increases both `L_QUANTITY` and `L_EXTENDEDPRICE` beyond the previous maximum values. Hence, a variable `min_value` obtains a value larger than `max_value` and the returned result `max_value-min_value` is negative. Maxim minus minimum never returns a negative value.

(2) (2 marks)
Rewrite a stored function such that it can be processed at `READ COMMITTED` level and show how the improved function interleaves the operations with another transaction and such that a result returned by the function is always correct.

```
CREATE OR REPLACE FUNCTION MAX_MIN ( orderkey IN NUMBER )
RETURN NUMBER IS
 max_min_value NUMBER(12);

BEGIN
 SELECT MAX(L_QUANTITY * L_EXTENDEDPRICE) -
        MIN(L_QUANTITY * L_EXTENDEDPRICE)
 INTO max_min_value
 FROM LINEITEM
 WHERE L_ORDERKEY = orderkey;

 RETURN max_min_value;
END MAX_MIN;
/
```

MAX-MIN transaction

```
EXECUTE MAX_MIN ( 1 )

SELECT MAX(L_QUANTITY * L_EXTENDEDPRICE) -
       MIN(L_QUANTITY * L_EXTENDEDPRICE)
INTO max_min_value
FROM LINEITEM
WHERE L_ORDERKEY = 1;

RETURN max_value-min_value;
```

Other transaction

```
UPDATE LINEITEM
SET L_QUANTITY =
       (SELECT MAX(L_QUANTITY)+1
        FROM LINEITEM),
    L_EXTENDEDPRICE =
       (SELECT MAX(L_EXTENDEDPRICE)+1
        FROM LINEIITEM;
WHERE L_ORDEKEY = 1;
COMMIT
```

MAX-MIN transaction always returns a correct value due to statement level consistency property and in a case given above it executes before Other transaction. Hence, a serial order is: MAX-MIN transaction before Other transaction.

When visualizing the concurrent executions use a technique of two-dimensional diagrams presented to you during the lecture classes, for example, see a presentation 14 Transaction Processing in Oracle DBMS slide 16.