Task 4 (2 marks)
Deadlocks

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

```
CREATE OR REPLACE PROCEDURE SWAP( order_key1  IN NUMBER,
                                  order_key2  In NUMBER ) IS

 total_price1 ORDERS.O_TOTALPRICE%TYPE;
 total_price2 ORDERS.O_TOTALPRICE%TYPE;

BEGIN
 SELECT O_TOTALPRICE
 INTO total_price1
 FROM ORDERS
 WHERE O_ORDERKEY = order_key1;

 SELECT O_TOTALPRICE
 INTO total_price2
 FROM ORDERS
 WHERE O_ORDERKEY = order_key2;

 UPDATE ORDERS
 SET O_TOTALPRICE = total_price1
 WHERE O_ORDERKEY = order_key2;

 UPDATE ORDERS
 SET O_TOTALPRICE = total_price2
 WHERE O_ORDERKEY = order_key1;

END SWAP;
/
```

Show a sample concurrent execution of two transactions both processing a function `SWAP`, both running at `READ COMMITTED` level, and such that the execution leads to a deadlock.

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.

Transaction 1	Transaction 2
SWAP(1,2)	SWAP(2,1)
SELECT O_TOTALPRICE INTO total_price1 FROM ORDERS WHERE O_ORDERKEY = 1;	
	SELECT O_TOTALPRICE INTO total_price1 FROM ORDERS WHERE O_ORDERKEY = 2;
	SELECT O_TOTALPRICE INTO total_price2 FROM ORDERS WHERE O_ORDERKEY = 1;
SELECT O_TOTALPRICE INTO total_price2 FROM ORDERS WHERE O_ORDERKEY = 2;	
UPDATE ORDERS SET O_TOTALPRICE = total_price1 WHERE O_ORDERKEY = 2;	
	UPDATE ORDERS SET O_TOTALPRICE = total_price1 WHERE O_ORDERKEY = 1;
UPDATE ORDERS SET O_TOTALPRICE = total_price2 WHERE O_ORDERKEY = 1; WAIT	
	UPDATE ORDERS SET O_TOTALPRICE = total_price1 WHERE O_ORDERKEY = 2; WAIT