#### Question 1

Oracle Database provides a function for returning the date of the last day of the month. It does not, however, provide a function for returning the date of the first day. Which of the following can be used to do this?

1. CREATE OR REPLACE FUNCTION plch\_first\_day (date\_in IN DATE)  
      RETURN DATE  
   IS  
   BEGIN  
      RETURN TRUNC (date\_in);  
   END;  
   /

1. CREATE OR REPLACE FUNCTION plch\_first\_day (date\_in IN DATE)  
      RETURN DATE  
   IS  
   BEGIN  
      RETURN TRUNC (date\_in, 'MM');  
   END;  
   /

1. CREATE OR REPLACE FUNCTION plch\_first\_day (date\_in IN DATE)  
      RETURN DATE  
   IS  
   BEGIN  
      RETURN TRUNC (date\_in, 'MONTH');  
   END;  
   /

1. CREATE OR REPLACE FUNCTION plch\_first\_day (date\_in IN DATE)  
      RETURN DATE  
   IS  
   BEGIN  
      RETURN TO\_DATE (TO\_CHAR (date\_in, 'YYYY-MM')   
   || '-01', 'YYYY-MM-DD');  
   END;  
   /

**Question 2**

Given this declaration section:

DECLARE  
   c\_format   CONSTANT VARCHAR2 (22)  
      := 'YYYY-MM-DD HH24:MI:SS' ;  
   l\_new\_year          DATE  
      := TO\_DATE (  
            '2012-01-02 00:00:01'  
          ,  c\_format);

which of the following blocks offers an exception section so that after that block is executed, the date and time 2012-01-01 00:00:01 will be displayed on the screen?

1. BEGIN  
      DBMS\_OUTPUT.put\_line (  
         TO\_CHAR (  
            l\_new\_year - 24  
          ,  c\_format));  
   END;
2. BEGIN  
      DBMS\_OUTPUT.put\_line (  
         TO\_CHAR (l\_new\_year - 1  
                ,  c\_format));  
   END;
3. BEGIN  
      DBMS\_OUTPUT.put\_line (  
         TO\_CHAR (  
              l\_new\_year  
            - 24 \* 60 \* 60  
          ,  c\_format));  
   END;
4. BEGIN  
      DBMS\_OUTPUT.put\_line (  
         TO\_CHAR (  
              TRUNC (l\_new\_year)  
            - 1  
            + 1 / (24 \* 60 \* 60)  
          ,  c\_format));  
   END;

**Question 1**

Which of these blocks will result in an unhandled ORA-00001 exception?

1. BEGIN
2. RAISE DUP\_VAL\_ON\_INDEX;
3. END;
4. /

1. BEGIN
2. RAISE -1;
3. END;
4. /

1. CREATE TABLE plch\_tab (n NUMBER PRIMARY KEY)
2. /
3. BEGIN
4. INSERT INTO plch\_tab
5. VALUES (1);
6. INSERT INTO plch\_tab
7. VALUES (1);
8. END;
9. /

1. BEGIN
2. RAISE DUP\_VAL\_ON\_INDEX;
3. EXCEPTION
4. WHEN OTHERS
5. THEN
6. RAISE;
7. END;
8. /

**Question 2**

Assume that the plch\_tab table has been created with a single numeric column. What change can I make in the following procedure so that it will compile without error?

CREATE OR REPLACE PROCEDURE plch\_proc (divisor\_in in NUMBER)

IS

BEGIN

INSERT INTO plch\_tab

VALUES (100/divisor\_in);

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX AND NO\_DATA\_FOUND

THEN

RAISE;

#### Records

I create and populate this table:

CREATE TABLE plch\_parts

(

partnum INTEGER PRIMARY KEY,

partname VARCHAR2 (100) UNIQUE

)

/

BEGIN

INSERT INTO plch\_parts

VALUES (100, 'Keyboard');

COMMIT;

END;

/

#### Question

Which choices contain code I can use in place of the /\*DECLARE\*/ comment in the following block so that after the resulting block executes, “Keyboard” will be displayed?

DECLARE

/\*DECLARE\*/

BEGIN

SELECT \*

INTO l\_part

FROM plch\_parts

WHERE partnum = 100;

DBMS\_OUTPUT.put\_line

(l\_part.partname);

END;

/

1. l\_part plch\_parts%TYPE;
2. \_part plch\_parts;
3. l\_part plch\_parts%ROWTYPE;
4. >CURSOR parts\_cur
5. IS
6. SELECT \* FROM plch\_parts;
7. l\_part parts\_cur%ROWTYPE;

I create and populate my employees table as follows:

CREATE TABLE plch\_employees

(

   employee\_id   INTEGER,

   last\_name     VARCHAR2 (100)

)

/

BEGIN

   INSERT INTO plch\_employees

        VALUES (100, ‘Picasso’);

   INSERT INTO plch\_employees

        VALUES (200, ‘Mondrian’);

   INSERT INTO plch\_employees

        VALUES (300, ‘O’’Keefe’);

   COMMIT;

END;

/

### Question

Which of these blocks will uppercase the last names of all employees in the table?

**a.**

DECLARE

   TYPE ids\_t IS TABLE OF plch\_employees.employee\_id%TYPE;

   l\_ids   ids\_t := ids\_t (100, 200, 300);

BEGIN

   FORALL indx IN 1 .. l\_ids.COUNT

   LOOP

      UPDATE plch\_employees

         SET last\_name = UPPER (last\_name)

       WHERE employee\_id = l\_ids (indx);

   END LOOP;

END;

/

**b.**

DECLARE

   TYPE ids\_t IS TABLE OF plch\_employees.employee\_id%TYPE;

   l\_ids   ids\_t := ids\_t (100, 200, 300);

BEGIN

   FORALL indx IN 1 .. l\_ids.COUNT

      UPDATE plch\_employees

         SET last\_name = UPPER (last\_name)

       WHERE employee\_id = l\_ids (indx);

END;

/

**c.**

BEGIN

   UPDATE plch\_employees

       SET last\_name = UPPER (last\_name);

END;

/

d.

 DECLARE

   TYPE ids\_t IS TABLE OF plch\_employees.employee\_id%TYPE;

   l\_ids   ids\_t := ids\_t (100, 200, 300);

BEGIN

   FORALL indx IN INDICES OF l\_ids

      UPDATE plch\_employees

         SET last\_name = UPPER (last\_name)

       WHERE employee\_id = l\_ids (indx);

END;

/

**Question**

[**http://www.oracle.com/technetwork/issue-archive/2013/13-mar/o23plsql-1906474.html**](http://www.oracle.com/technetwork/issue-archive/2013/13-mar/o23plsql-1906474.html)

The plch\_employees table has a single-column primary key named employee\_id. I need to write a function with the following header: FUNCTION plch\_one\_employee (

employee\_id\_in IN PLS\_INTEGER)

RETURN plch\_employees%ROWTYPE

that returns a record that contains the row of information for the specified primary key. Which of the choices provides an implementation of PLCH\_ONE\_EMPLOYEE that meets this requirement?

**a.**

IS

l\_return plch\_employees%ROWTYPE;

BEGIN

SELECT \*

INTO l\_return

FROM plch\_employees

WHERE employee\_id = employee\_id\_in;

RETURN l\_return;

END plch\_one\_employee;

**b.**

IS

CURSOR one\_emp\_cur

IS

SELECT \*

FROM plch\_employees

WHERE employee\_id = employee\_id\_in;

l\_return one\_emp\_cur%ROWTYPE;

BEGIN

OPEN one\_emp\_cur;

FETCH one\_emp\_cur INTO l\_return;

CLOSE one\_emp\_cur;

RETURN l\_return;

END plch\_one\_employee;

**c.**

IS

l\_return plch\_employees%ROWTYPE;

BEGIN

FOR rec IN (SELECT \*

FROM plch\_employees

WHERE employee\_id = employee\_id\_in)

LOOP

l\_return := rec;

END LOOP;

RETURN l\_return;

END plch\_one\_employee;

**d.**

IS

l\_cursor SYS\_REFCURSOR;

l\_return plch\_employees%ROWTYPE;

BEGIN

OPEN l\_cursor FOR

SELECT \*

FROM plch\_employees

WHERE employee\_id = employee\_id\_in;

FETCH l\_cursor INTO l\_return;

CLOSE l\_cursor;

RETURN l\_return;

END plch\_one\_employee;

**Question**

Which of the following blocks will display these three lines after execution:

<http://www.oracle.com/technetwork/issue-archive/2012/12-jul/o42plsql-1653077.html>

Strawberry

Raspberry

Blackberry

**a.**

DECLARE

l\_names DBMS\_UTILITY.maxname\_array;

BEGIN

l\_names (1) := 'Strawberry';

l\_names (10) := 'Blackberry';

l\_names (2) := 'Raspberry';

FOR indx IN 1 .. l\_names.COUNT

LOOP

DBMS\_OUTPUT.put\_line (l\_names (indx));

END LOOP;

END;

/

**b.**

DECLARE

l\_names DBMS\_UTILITY.maxname\_array;

BEGIN

l\_names (1) := 'Strawberry';

l\_names (10) := 'Blackberry';

l\_names (2) := 'Raspberry';

indx := l\_names.FIRST;

WHILE (indx IS NOT NULL)

LOOP

DBMS\_OUTPUT.put\_line (l\_names (indx));

indx := l\_names.NEXT (indx);

END LOOP;

END;

/

**c.**

DECLARE

l\_names DBMS\_UTILITY.maxname\_array;

BEGIN

l\_names (1) := 'Strawberry';

l\_names (10) := 'Blackberry';

l\_names (2) := 'Raspberry';

DECLARE

indx PLS\_INTEGER := l\_names.FIRST;

BEGIN

WHILE (indx IS NOT NULL)

LOOP

DBMS\_OUTPUT.put\_line (l\_names (indx));

indx := l\_names.NEXT (indx);

END LOOP;

END;

END;

/

**d.**

DECLARE

l\_names DBMS\_UTILITY.maxname\_array;

BEGIN

l\_names (1) := 'Strawberry';

l\_names (10) := 'Blackberry';

l\_names (2) := 'Raspberry';

FOR indx IN l\_names.FIRST .. l\_names.LAST

LOOP

DBMS\_OUTPUT.put\_line (l\_names (indx));

END LOOP;

END;

/