1. Create an associative array with the element type of a user-defined record. This record should contain the first name, last name, and total number of courses that a particular instructor teaches. Display the records of the associative array on the screen.

SOLUTION:

```
SET SERVEROUTPUT ON
DECLARE
CURSOR instructor_cursor IS
SELECT i.first_name, i.last_name, COUNT(UNIQUE s.course_no) t_courses
FROM instructor i, section s
WHERE i.instructor_id = s.instructor_id
GROUP BY i.first_name, i.last_name;

TYPE instructor_record_type IS RECORD
(first_name VARCHAR2(25),
last_name VARCHAR2(25),
total_courses NUMBER(2) );

TYPE instructor_type IS TABLE OF instructor_record_type
INDEX BY BINARY_INTEGER;

instructor_tab instructor_type;
v_counter INTEGER := 0;
BEGIN
FOR instructor_record in instructor_cursor LOOP
    v_counter := v_counter + 1;
    instructor_tab(v_counter).first_name := instructor_record.first_name;
    instructor_tab(v_counter).last_name := instructor_record.last_name;
    instructor_tab(v_counter).total_courses := instructor_record.t_courses;

    DBMS_OUTPUT.PUT_LINE(instructor_tab(v_counter).first_name||' '
    ||instructor_tab(v_counter).last_name||' '||
    'has'||' '||instructor_tab(v_counter).total_courses||' courses!');

END LOOP;
END;
```

OUTPUT:

```
Anca Pup has 2 courses!
Alexa Iuga has 3 courses!
Andrei Opra has 2 courses!
```

2. Modify the script you just created. Instead of using an associative array, use a nested table.

SOLUTION:

```
SET SERVEROUTPUT ON
DECLARE
CURSOR instructor_cursor IS
SELECT i.first_name, i.last_name, COUNT(UNIQUE s.course_no) t_courses
FROM instructor i, section s
WHERE i.instructor_id = s.instructor_id
GROUP BY i.first_name, i.last_name;

TYPE instructor_record_type IS RECORD
(first_name VARCHAR2(25),
last_name VARCHAR2(25),
total_courses NUMBER(2) );

TYPE instructor_type IS TABLE OF instructor_record_type;

instructor_tab instructor_type := instructor_type();
v_counter INTEGER := 0;
BEGIN
FOR instructor_record in instructor_cursor LOOP
   v_counter := v_counter + 1;
   instructor_tab.EXTEND;
   instructor_tab(v_counter).first_name := instructor_record.first_name;
   instructor_tab(v_counter).last_name := instructor_record.last_name;
   instructor_tab(v_counter).total_courses := instructor_record.t_courses;

   DBMS_OUTPUT.PUT_LINE(instructor_tab(v_counter).first_name||' '
   ||instructor_tab(v_counter).last_name||' '||
   'has'||' '||instructor_tab(v_counter).total_courses||' courses!');

END LOOP;
END;
```

OUTPUT:

```
Anca Pup has 2 courses!
Alexa Iuga has 3 courses!
Andrei Opra has 2 courses!
```

3. Modify the script you just created. Instead of using a nested table, use a varray.

SOLUTION:

```
SET SERVEROUTPUT ON
DECLARE
CURSOR instructor_cursor IS
SELECT i.first_name, i.last_name, COUNT(UNIQUE s.course_no) t_courses
FROM instructor i, section s
WHERE i.instructor_id = s.instructor_id
GROUP BY i.first_name, i.last_name;

TYPE instructor_record_type IS RECORD
(first_name VARCHAR2(25),
last_name VARCHAR2(25),
total_courses NUMBER(2) );

TYPE instructor_type IS VARRAY(50) OF instructor_record_type;

instructor_varray instructor_type := instructor_type();
v_counter INTEGER := 0;
BEGIN
FOR instructor_record in instructor_cursor LOOP
    v_counter := v_counter + 1;
    instructor_varray.EXTEND;
    instructor_varray(v_counter).first_name := instructor_record.first_name;
    instructor_varray(v_counter).last_name := instructor_record.last_name;
    instructor_varray(v_counter).total_courses := instructor_record.t_courses;

    DBMS_OUTPUT.PUT_LINE(instructor_varray(v_counter).first_name||' '
    ||instructor_varray(v_counter).last_name||' '||
    'has'||' '||instructor_varray(v_counter).total_courses||' courses!');

END LOOP;
END;
```

OUTPUT:

```
Anca Pup has 2 courses!
Alexa Iuga has 3 courses!
Andrei Opra has 2 courses!
```

4. Create a user-defined record with four fields: course_no, description, cost, and prerequisite_rec. The last field, prerequisite_rec, should be a user-defined record with three fields: prereq_no, prereq_desc, and prereq_cost. For any ten courses that have a prerequisite course, populate the user-defined record with all the corresponding data, and display its information on the screen.

SOLUTION:

```
SET SERVEROUTPUT ON
DECLARE
CURSOR course_cursor IS
SELECT course_no, description, cost, prerequisite
FROM course
WHERE prerequisite IS NOT NULL AND rownum <= 10;

TYPE prerequisite_rec_record_type IS RECORD
(
prereq_no course.prerequisite%TYPE,
prereq_desc course.description%TYPE,
prereq_cost course.cost%TYPE
);

TYPE course_record_type IS RECORD
(course_no NUMBER(38),
description VARCHAR2(50),
cost NUMBER(9,2),
prerequisite_rec prerequisite_rec_record_type
);

course_record course_record_type;

BEGIN
   FOR course_rec IN course_cursor LOOP
      course_record.course_no := course_rec.course_no;
      course_record.description := course_rec.description;
      course_record.cost := course_rec.cost;

      SELECT course_no, description, cost
      INTO course_record.prerequisite_rec.prereq_no,
          course_record.prerequisite_rec.prereq_desc,
          course_record.prerequisite_rec.prereq_cost

      FROM course
      WHERE course_no = course_rec.prerequisite;

      DBMS_OUTPUT.PUT_LINE('Course with ID: '||course_record.course_no|| ' is: '
        ||course_record.description|| ' costs: '||course_record.cost||' require subjects: '
        ||course_record.prerequisite_rec.prereq_no);
    END LOOP;
END;
```

OUTPUT: