**TASK 1.**

* 1. Write PL-SQL program, which prints multipliers of number 9 i.e., 9,18,27, …, up to 30\*9
  2. Prepare PL/SQL procedure which does the same as the task 1. Run that procedure.
  3. Write a function which returns the greatest common divisor (highest common factor) of two numbers.
  4. Prepare a function which displays n-th number in a Fibonacci sequence (1,1,2,3,5,8,13...) use recursion, then do it without using recursion.
  5. Create a table with the name »TEST« with columns »NAME« and »SURNAME« and write procedure, that inserts data to it. Parameters should be p\_name and p\_surname. Call procedure and check if data was added to the table.
  6. Change procedure from task 5. in a way that would not allow to insert the same person multiple times
  7. Add unique key to the table, so it will not be possible to insert one person more than one time
  8. Study which types are supported in oracle database. From now on use only the following:

NUMBER for all numeric columns,

VARCHAR2(size) for all character columns and

DATE for all date columns.

It is much easier if we don't over-complicate our lives.

**TASK 2.**

2.1 Use constants in task 1.1

2.2 Create table with the same columns as »TEST« but change a name to make it more meaningful. Add column id, which should be used as primary key, create sequence that you will insert when adding the data. Adapt procedure from task 1.6 in a way that it will insert name, surname and sequence value. The data should be inserted in new table, of course.

2.3 Create table with the information of the presence in the job. Table should have meaningful name, the columns should include the data about: Date, Number of hours, Comment. The table from task 2.2 should be connected using the field id. Define foreign key between those two tables.

2.4 Add to the table from task 2.3 a column which hold information about regular hours/overtime hours. Create new table with this information (hour types) and create foreign key between them.

2.5 Create procedure, which inserts data to table from task 2.3. Parameters should be id of a person, number of hours and hour type.

* 1. Create procedure, which inserts data to table from task 2.3. Parameters should be id of a person, work start date, work end date (date columns in oracle include data about hours, minutes and seconds also). Your procedure should calculate number of hours from provided dates. If start date and end date are different (not in the same day) insert one record for each day (calculate and insert the appropriate number of hours). Use procedure from task 2.5 to insert one record. A person can for each day have 8 regular hours only, the rest should be overtime hours.
  2. Prepare database schema (Create tables, primary keys, foreign keys) to store information of employees and cars of a company. Include data about car manufacturer, car model, car manufacturing year, color, equipment package, registration number, add a column for comment. Use master/detail as you find meaningful. It should be possible for one employee to have multiple cars. Schema should also include data about history of car registrations.

**ANSWERS:**

**Task 1.**

DECLARE

i NUMBER := 1;

BEGIN

LOOP

EXIT WHEN i> 30 ;

DBMS\_OUTPUT.PUT\_LINE(9 \* i);

i := i+1;

END LOOP;

END;

**2a. Package specification syntax**

CREATE OR REPLACE

PACKAGE TASK1 AS

procedure Q2;

END TASK1;

*(press compile)*

**2b. Package body syntax**

CREATE OR REPLACE PACKAGE BODY TASK1 AS

procedure Q2 AS

i NUMBER := 1;

BEGIN

LOOP

EXIT WHEN i> 30 ;

DBMS\_OUTPUT.PUT\_LINE(9 \* i);

i := i+1;

END LOOP;

END Q2;

END TASK1;

*(press compile)*

**2c. call procedure syntax**

BEGIN

TASK1.Q2;

END;

**3a. package specification syntax**

create or replace PACKAGE TASK1 AS

function Q3 (p\_num1 IN number, p\_num2 IN number) return number;

END TASK1;

**3b. Package body syntax**

create or replace PACKAGE BODY TASK1 AS

function Q3 (p\_num1 IN number, p\_num2 IN number) return number AS

-- declare variable num1, num2 and temp

num1 NUMBER:=p\_num1;

num2 NUMBER:=p\_num2;

temp NUMBER;

BEGIN

WHILE MOD(num2, num1) != 0 LOOP

temp := MOD(num2, num1);

num2 := num1;

num1 := temp;

END LOOP;

RETURN num1;

END Q3;

END TASK1;

**3c. call function syntax**

select task1.Q3(42,98) from dual;

**1.4 Recursive**

function Q4recursion (k IN number) RETURN number AS

BEGIN

if k = 1 then

return 0;

elsif k = 2 then

return 1;

else

return Q4recursion(k - 2) + Q4recursion(k - 1);

end if;

END Q4recursion;

**Non-recursive**

function Q4nonrecursion (p\_kth IN number) RETURN number AS

num1 NUMBER:= 0;

num2 NUMBER:= 1;

kth NUMBER := p\_kth;

temp NUMBER := 0;

nsum NUMBER := 0;

BEGIN

while kth > temp LOOP

nsum := nsum + num1;

num1 := num2;

num2 := nsum;

temp := temp+1;

END LOOP;

RETURN nsum;

END Q4nonrecursion;

**1.5**

procedure Q5(

p\_name IN TEST.NAME%TYPE,

p\_surname IN TEST.SURNAME%TYPE)IS

BEGIN

INSERT INTO TEST ("NAME", "SURNAME")

VALUES (p\_name, p\_surname);

COMMIT;

END Q5;

**1.6**

procedure Q6 (

p\_name IN TEST.NAME%TYPE,

p\_surname IN TEST.SURNAME%TYPE)IS

BEGIN

INSERT INTO TEST ("NAME", "SURNAME")

SELECT p\_name, p\_surname

FROM TEST

WHERE NOT EXISTS

(

SELECT 1 FROM TEST WHERE "NAME" = p\_name AND "SURNAME" = p\_surname

);

COMMIT;

END Q6;