**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.

**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;