ACTIVIDADES TEMA 6 PARTE II PL/SQL - SUBPROGRAMAS SOLUCIONES

1. Escribir un script que permite calcular el doble y la mitad de un número

```
CREATE OR REPLACE PROCEDURE subprog_ejer2 (num number)
IS
      doble number(6);
      mitad number(6,2);

BEGIN
   doble:=num*2;
   mitad:= num/2;
   dbms_output.put_line( 'Numero: '|| num|| '   Doble: '||doble||'    Mitad: '||mitad);
END subprog_ejer2;
```

Prueba a realizar ahora el mismo ejemplo utilizando procedimientos o funciones individuales para cada uno de ellos. Realiza el bloque para comprobar que funcionan.

2. Codificar un procedimiento que reciba una cadena y la visualice al revés.

```
CREATE OR REPLACE PROCEDURE p_cadena_reves(vcadena VARCHAR2)
AS
      vcad_reves VARCHAR2(80);
BEGIN
      FOR i IN REVERSE 1..LENGTH(vcadena) LOOP
            vcad_reves := vcad_reves || SUBSTR(vcadena,i,1);
      END LOOP;

      DBMS_OUTPUT.PUT_LINE(vcad_reves);
END p_cadena_reves;
```

3. Codificar una función que reciba una cadena y devuelva la cadena al revés.

```
CREATE OR REPLACE FUNCTION f_cadena_reves(vcadena varchar2) RETURN varchar2 IS
         vcad_reves varchar2(50);

BEGIN

      FOR i IN REVERSE 1..LENGTH(vcadena) LOOP
            vcad_reves := vcad_reves || SUBSTR(vcadena,i,1);
      END LOOP;

      return vcad_reves;

END f_cadena_reves;
```

4. ***Crear un bloque que utilice el procedimiento y la función creados anteriormente para visualizar la cadena al reves.***

```
declare
      rtdo  varchar2(20);
      cad         varchar2(50);
begin
      cad    :='Hola mundo';
      DBMS_OUTPUT.PUT_LINE('La cadena es :'||cad);

      -- usando el procedimiento
      p_cadena_reves(vcadena VARCHAR2);
      --en nuestro caso el propio procedimiento recibe la cadena y la visualiza

      -- usando la función
      rtdo := f_cadena_reves(cad);   -- la función devuelve la cadena al revés
      DBMS_OUTPUT.PUT_LINE('La cadena al reves es usando la función es :'||rtdo);
end;
```

5. ***Escribir un script que una vez introducido el código de departamento visualice utilizando un procedimiento el número de empleados de dicho departamento.***

```
CREATE OR REPLACE PROCEDURE total_emp_por_dep (p_dept number)
IS
  v_total number;
BEGIN
   select count(*) into v_total from emple where dept_no = p_dept;
   dbms_output.put_line('El numero de empleados en el dept_no: '|| p_dept ||' es '||
v_total);

END total_emp_por_dep;

====================

accept v_dept prompt 'Introduce el numero de depart a consultar: ';

begin
       dbms_output.put_line('Llamamos al procedimiento que calcula el número de
empleados en el departamento '||&v_dept);
       total_emp_por_dep (&v_dept);
end;
```

6. ***Crear un procedimientos que permita visualizar todos los datos de un usuario a partir de su número de empleado, visualizar además el nombre de departamento al que pertenece***

```
CREATE OR REPLACE PROCEDURE ver_emp_por_empno (v_emp number)
IS
  v_reg             emple%rowtype;
  v_dnombre         depart.dnombre%type;

BEGIN
   select * into v_reg from emple where emp_no = v_emp;
   select dnombre into v_dnombre from depart where dept_no= v_reg.dept_no;
   dbms_output.put_line('num empleado    :'|| v_emp );
   dbms_output.put_line('Nombre          :'|| v_reg.apellido );
```

```
   dbms_output.put_line('Oficio          :'|| v_reg.oficio );
   dbms_output.put_line('Su Jefe         :'|| v_reg.dir );
   dbms_output.put_line('Salario         :'|| v_reg.salario);
   dbms_output.put_line('Comision        :'|| v_reg.comision);
   dbms_output.put_line('Departamento    :'|| v_reg.dept_no ||'   ' ||v_dnombre);
END ver_emp_por_empno;
```

7. ***Realizar un bloque que una vez introducido por teclado el código de un empleado, visualice los datos del empleado utilizando el procedimiento anterior.***

```
accept v_emp_no prompt 'Introduce el numero del empleado a consultar: ';

begin
       dbms_output.put_line('Llamamos al procedimiento que visualiza los datos
                  del empleado '||&v_emp_no);
       ver_emp_por_empno (v_emp);
end;
```

8. ***Crear un procedimiento que permita calcular la nómina de un empleado y visualizarla***

```
CREATE OR REPLACE PROCEDURE calc_nomina (v_empno emple.emp_no%type)
IS
  v_nomina          number;
  v_reg                  emple%rowtype;
BEGIN
   select * into v_reg from emple where emp_no= v_empno;
   v_nomina:= v_reg.salario + nvl(v_reg.comision,0);
   dbms_output.put_line('La nomina del empleado '|| v_empno ||' de nombre '||
                        v_reg.apellido || ' es '|| v_nomina);
END calc_nomina;
```

9. ***Realizar un bloque que una vez introducido por teclado el código de un empleado, visualice la nómina que cobrará en empleado utilizando el procedimiento anterior.***

```
accept v_emp_no prompt 'Introduce el numero del empleado para calcular su nomima: ';

begin
       dbms_output.put_line('Llamamos al procedimiento que calcula la nomina
                         del empleado '||&v_emp_no);
       calc_nomina (v_emp);
end;
```

10. ***Diseñar una función llamada FUNC_NUM_EMP que permita devolver el emp_no de un determinado empleado a partir de su apellido, el departamento y el oficio que tiene. Diseñar después el bloque para comprobarlo.***

```
CREATE OR REPLACE FUNCTION devuelve_empno (p_apel varchar2, p_dep number, p_oficio
varchar2) RETURN emple.emp_no%type
IS
  v_empno emple.emp_no%type;
BEGIN
   -- para evitar problemas de mayúsculas o minúsculas usamos el UPPER
```

```
   select emp_no into v_empno from emple
             where UPPER(apellido)=UPPER(p_apel) and UPPER(oficio)=UPPER(p_oficio)
                       and dept_no= p_dep;

   return v_empno;
END devuelve_empno;

---

--Se supone que los datos se introducirán correctamente
accept p_apel prompt 'Introduce el apellido    :';
accept p_dep  prompt 'Introduce el numero de departamento :';
accept p_oficio prompt 'Introduce el oficio :';

declare
  num_emp emple.emp_no%type;
begin

       num_emp := devuelve_empno ('&p_apel', &p_dep , '&p_oficio');
       dbms_output.put_line ('El numero de empleado de '|| '&p_apel' ||
            'del departamento ' ||&p_dep||' y de oficio '|| '&p_oficio'||' es '||
      num_emp);
end;
```

11. Diseñar una función recursiva que permita calcular el término enésimo de la sucesión de Fibonacci, sabiendo que los dos primeros términos son 1 y que cualquier otro término se calcula como la suma de los dos anteriores

1, 1, 2, 3, 5, 8, 13, 21,…

```
CREATE OR REPLACE FUNCTION fibonacci (pos number) RETURN NUMBER
IS
  pos1 number;
  pos2 number;
  valor number;

BEGIN
  if pos<=2 then
        valor:=1;
      else
         valor :=fibonacci(pos-1) + fibonacci(pos-2);
  end if;

  return valor;

END fibonacci;
```

12. Diseñar un bloque de programa que introducida una posición, llame a la función anterior y nos muestre el elemento enésimo de la sucesión de fibonacci

```
accept pos prompt 'Introduce la posicion a calcular';
declare
   elemento number;
begin
   if (&pos < 3) then
     dbms_output.put_line('los primero elementos de la sucesion de fibonacci son 1 1');
```

```
      else
     elemento:= fibonacci(&pos - 1) + fibonacci(&pos- 2);
       dbms_output.put_line('El elemento '||&pos|| ' de la succesion de fibonacci es '||
elemento);
      end if;
end;
```

13. Escribir un bloque que permita escribir los 'n' primeros términos de la sucesión de fibonacci

```
accept num_term prompt 'Introduce la posicion a calcular';
declare
   elemento number;
begin
 for pos in 1..&num_term loop
   if (pos < 3) then
     dbms_output.put('1 ');
      else
     elemento:= fibonacci(pos - 1) + fibonacci(pos - 2);
       dbms_output.put(elemento||' ');
      end if;
  end loop;
  dbms_output.put_line(' ');
end;
```