Tarea 10: Corto 1

Angela Beatriz Canel Hernández, 201906569
Escuela de Mecánica Eléctrica, Facultad de Ingeniería, Universidad de San Carlos

***Resumen*— A continuación, se exponen catorce programas implementados en Octave, cada uno destinado a aplicaciones específicas. Cada programa establece una conexión con una tabla específica en una base de datos PostgreSQL, y en cada iteración de ejecución, incorpora nuevos datos a la base de datos correspondiente.**

I. GITHUB

https://github.com/ac428/Corto1-proyectos-IE

II. CÓDIGO

A. Problema 1

Pedir 3 números: Si el primero es el más grande mostrar la suma de los tres números. Si el segundo es el más grande mostrar la multiplicación de los 3 números. Si el tercero es el más grande concatenar los 3 numero. Si hay dos iguales mostrar el único que no es igual. Si los tres son iguales mostrar los números y el mensaje: *Todos son iguales".

```
%PROBLEMA 1 -- SUMA, MULTIPLICACIÓN O CONCATENACIÓN DE TRES NÚMEROS
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Crear tabla en base de datos
%tablename = 'problema1';
%query = {'CREATE TABLE IF NOT EXISTS ', tablename, ' (num1 INT, num2 INT, num3 INT, resultado TEXT, operacion TEXT);'};
%pq_exec_params(conn, query);

% Pedir al usuario que ingrese 3 números
num1 = input("Ingrese el primer número: ");
num2 = input("Ingrese el segundo número: ");
num3 = input("Ingrese el tercer número: ");

% Verificar condiciones y realizar operaciones
if (num1 > num2) && (num1 > num3)
    % Si el primero es el más grande, mostrar la suma
    resultado = num1 + num2 + num3;
    operacion = 'Suma';
elseif (num2 > num1) && (num2 > num3)
    % Si el segundo es el más grande, mostrar la multiplicación
    resultado = num1 * num2 * num3;
    operacion = 'Multiplicación';
elseif (num3 > num1) && (num3 > num2)
    % Si el tercero es el más grande, concatenar los números
    resultado = strcat(num2str(num1), num2str(num2),num2str(num3));
    operacion = 'Concatenación';

elseif (num1 == num2) && (num2 == num3)
    % Si los tres son iguales, mostrar los números y un mensaje
    resultado = [num1, num2, num3];
    operacion = 'Todos son iguales';
else
    % Si hay dos iguales, mostrar el único que no es igual
    if num1 == num2
        resultado = num3;
    elseif num1 == num3
        resultado = num2;
    else
        resultado = num1;
    end
    operacion = 'Número único';
end

% Mostrar resultado
fprintf('Resultado de la operación (%s): %s\n', operacion, num2str(resultado));

% Insertar información en la tabla
query = {'INSERT INTO ', tablename, ' (num1, num2, num3, resultado, operacion) VALUES (', num2str(num1), ', ', num2str(num2), ', ', num2str(num3), ', ''', num2str(resultado), ''', ''', operacion, '''
pq_exec_params(conn, query);

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 1: Código en Octave

B. Problema 2

Pedir un número y mostrar sus divisores (valores que dividen al número en partes exactas).

```
%PROBLEMA 2 -- DIVISORES
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Crear tabla en base de datos
%tablename = 'problema2';
%query = {'CREATE TABLE IF NOT EXISTS ', tablename, ' (num INT, divisores TEXT, operacion TEXT);'};
%pq_exec_params(conn, query);

% Solicitar el número
num = input('Ingrese un número: ');

% Encontrar los divisores del número
divisors = [];
for i = 1:num
    if mod(num, i) == 0
        divisors = [divisors, i];
    end
end

% Mostrar los divisores en consola
disp(['Los divisores de ', num2str(num), ' son:']);
disp(divisors);

% Insertar información en la tabla
query = {'INSERT INTO ', tablename, ' (num, divisores, operacion) VALUES (', num2str(num), ', ''', strcat(num2str(divisors), ' '), ''', ''Divisores'');'};
pq_exec_params(conn, query);

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 2: Código en Octave

C. Problema 3

Pedir una palabra y contar cuantas vocales tiene. Ejemplo: Ingreso: palabra, Resultado: la palabra tiene 3 vocales.

```
%PROBLEMA 3 -- VOCALES
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Crear tabla en base de datos
%tablename = 'problema3';
%query = {'CREATE TABLE IF NOT EXISTS ', tablename, ' (word TEXT, num_vowels INT, operacion TEXT);'};
%pq_exec_params(conn, query);

% Solicitar la palabra
word = input('Ingrese una palabra: ', 's');

% Convertir la palabra a minúsculas
word = lower(word);

% Contar cuántas vocales tiene la palabra
numVowels = 0;
for i = 1:length(word)
    if ismember(word(i), 'aeiou')
        numVowels = numVowels + 1;
    end
end

% Mostrar cuántas vocales tiene la palabra en consola
disp(['La palabra ', word, ' tiene ', num2str(numVowels), ' vocales.']);

% Insertar la información en la tabla "problema3"
query = {'INSERT INTO ', tablename, ' (word, num_vowels, operacion) VALUES (''', word, ''', ', num2str(numVowels), ', ''Vocales'');'};
pq_exec_params(conn, query);

% Cerrar la conexión con la base de datos
```

Figure 3: Código en Octave

D. Problema 4

Pedir un número y mostrar la suma de los números desde 0 hasta ese número:
Ejemplo:
Ingreso 6 (debe sumar 1+2+3+4+5+6) Resultado: 21

```
%PROBLEMA 4 -- suma de números
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Crear tabla en base de datos
%tablename = 'problema4';
%query = {'CREATE TABLE ', tablename, ' (num INT, sum INT, operacion TEXT);'};
%pq_exec_params(conn, query);


% Solicitar un número
num = input('Ingrese un número: ');

% Calcular la suma de los números desde 0 hasta el número ingresado
sum = 0;
for i = 0:num
    sum = sum + i;
end

% Mostrar la suma en la consola
disp(['La suma de los números desde 0 hasta ', num2str(num), ' es: ', num2str(sum)]);

% Insertar la información en la tabla "problema4"
query = {'INSERT INTO ', tablename, ' (num, sum, operacion) VALUES (', num2str(num), ', ', num2str(sum), ', ''Suma'');'};
pq_exec_params(conn, query);

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 4: Código en Octave

E. Problema 5

Pedir un número de inicio y un número de fin y mostrar los números de 2 en 2 desde el número de inicio hasta el número de fin.
Ejemplo:
Ingreso 5 y 16, Resultado: 5, 7, 9, 11, 13, 15
Ingreso 6 y 12, Resultado: 6, 8, 10, 12

```
%PROBLEMA 5 -- NUMEROS DE DOS EN DOS
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Crear tabla en base de datos
%tablename = 'problema5';
%query = ['CREATE TABLE ', tablename, ' (num INT, num_inicio INT, num_fin INT, operacion TEXT);'];
%pq_exec_params(conn, query);


% Solicitar un número de inicio y un número de fin
num_inicio = input('Ingrese un número de inicio: ');
num_fin = input('Ingrese un número de fin: ');

% Calcular los números de 2 en 2 desde el número de inicio hasta el número de fin
for i = num_inicio:2:num_fin
    % Mostrar los números en la consola
    disp(i);

    % Insertar la información en la tabla "problema5"
    query = ['INSERT INTO ', tablename, ' (num, num_inicio, num_fin, operacion) VALUES (', num2str(i), ', ', num2str(num_inicio), ', ', num2str(num_fin), ', ''Numeros de 2 en 2'');'];
    pq_exec_params(conn, query);
end

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 5: Código en Octave

F. Problema 6

Pedir dos números, verificar cual es el mayor y mostrar la lista de números desde el mayor hasta el menor.
Ejemplo:
Ingreso: 3 y 9, Resultado: 9,8,7, 6,5, 4, 3
Ingreso: 10 y 7, Resultado: 10, 9, 8, 7

```
%PROBLEMA 6 -- LISTA DE MAYOR A MENOR
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Crear tabla en base de datos
%tablename = 'problema6';
%query = ['CREATE TABLE ', tablename, ' (num INT, num_inicio INT, num_fin INT, operacion TEXT);'];
%pq_exec_params(conn, query);


% Solicitar dos números
num1 = input('Ingrese el primer número: ');
num2 = input('Ingrese el segundo número: ');

% Verificar cual es el mayor y asignarlo a "num_mayor"
num_mayor = max(num1, num2);

% Calcular los números desde el número mayor hasta el número menor
for i = num_mayor:-1:min(num1, num2)
    disp(i); % Mostrar los números en la consola
end

% Insertar la información en la tabla "problema6"
query = ['INSERT INTO ', tablename, ' (num, num_inicio, num_fin, operacion) VALUES (', num2str(i), ', ', num2str(num1), ', ', num2str(num2), ', ''Lista de mayor a menor'');'];
pq_exec_params(conn, query);

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 6: Código en Octave

G. Problema 7

Pedir una palabra y contar cuantas veces aparece cada vocal.
Ejemplo:
Ingreso: Programación, Resultado: A=2, E=0, I=1, O=2, U=0

```
%PROGRAMA 7 -- CONTANDO LAS VOCALES
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Crear tabla en base de datos
%tablename = 'problema7';
%query = ['CREATE TABLE ', tablename, ' (palabra TEXT, cont_A INT, cont_E INT, cont_I INT, cont_O INT, cont_U INT);'];
%pq_exec_params(conn, query);

% Solicitar una palabra
palabra = input('Ingrese una palabra: ', 's');

% Convertir la palabra a minúsculas para contar las vocales de manera insensible a mayúsculas
palabra = lower(palabra);

% Inicializar contadores para cada vocal
cont_A = 0;
cont_E = 0;
cont_I = 0;
cont_O = 0;
cont_U = 0;
```

```
% Contar las ocurrencias de cada vocal en la palabra
for i = 1:length(palabra)
    switch palabra(i)
        case 'a'
            cont_A = cont_A + 1;
        case 'e'
            cont_E = cont_E + 1;
        case 'i'
            cont_I = cont_I + 1;
        case 'o'
            cont_O = cont_O + 1;
        case 'u'
            cont_U = cont_U + 1;
    end
end

% Mostrar el resultado
fprintf('A=%d, E=%d, I=%d, O=%d, U=%d\n', cont_A, cont_E, cont_I, cont_O, cont_U);

% Insertar la información en la tabla "problema7"
query = ['INSERT INTO ', tablename, ' (palabra, cont_A, cont_E, cont_I, cont_O, cont_U) VALUES (''', palabra, ''', ', num2str(cont_A), ', ', num2str(cont_E), ', ', num2str(cont_I), ',
pq_exec_params(conn, query);

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 7: Código en Octave

H. Problema 8

Programa que muestra los números impares desde el 1 hasta el 100 e indica cuantos número impares hay. Almacenar el resultado y opción de mostrar el historial.

```
% PROGRAMA 8 -- MOSTRAR NÚMEROS IMPARES Y CUANTOS HAY
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Crear tabla en base de datos
%tablename = 'problema8';
%query = ['CREATE TABLE ', tablename, ' (numero INT, opcion VARCHAR(20));'];
%pq_exec_params(conn, query);

% Crear lista para almacenar los números impares y la cantidad de números impares
impares = [];
cantidad = 0;

% Mostrar los números impares desde el 1 hasta el 100
for i = 1:100
    if mod(i, 2) == 0
        impares = [impares, i];
        fprintf('%d ', i);
        cantidad = cantidad + 1;
    end
end

fprintf('\nCantidad de números impares: %d\n', cantidad);
```

```
% Preguntar al usuario si desea guardar el resultado en la base de datos
save = input('¿Desea guardar el resultado en la base de datos? (S/N): ', 's');

if strcmpi(save, 'S') || strcmpi(save, 's')
    % Almacenar el resultado en la tabla "problema8"
    for i = 1:length(impares)
        query = ['INSERT INTO ', tablename, ' (numero, opcion) VALUES (', num2str(impares(i)), ', ', "'impares'", ');'];
        pq_exec_params(conn, query);
    end

    query = ['INSERT INTO ', tablename, ' (numero, opcion) VALUES (', num2str(cantidad), ', ', "'cantidad'", ');'];
    pq_exec_params(conn, query);

    fprintf('Resultados guardados en la base de datos.\n');
else
    fprintf('Resultados no guardados en la base de datos.\n');
end

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 8: Código en Octave

I. Problema 9

Programa que reciba tres números entrenos positivos, correspondiente a los tres lados del triángulo, debe mostrar si es Equilátero, isósceles o escaleno. Almacenar el resultado y opción de mostrar el historial.

```
% PROGRAMA 9 -- JUEGO DEL TRIÁNGULO

%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Solicitar al usuario que ingrese los tres lados del triángulo
a = input('Ingrese el primer lado del triángulo: ');
b = input('Ingrese el segundo lado del triángulo: ');
c = input('Ingrese el tercer lado del triángulo: ');

% Verificar si el triángulo es válido
if a + b <= c || a + c <= b || b + c <= a
    fprintf('Los tres números ingresados no forman un triángulo válido.\n');
else
    % Determinar el tipo de triángulo
    if a == b && b == c
        fprintf('El triángulo es equilátero.\n');
        resultado = 'equilátero';
    elseif a == b || a == c || b == c
        fprintf('El triángulo es isósceles.\n');
        resultado = 'isósceles';
    else
        fprintf('El triángulo es escaleno.\n');
        resultado = 'escaleno';
    end
end
```

```
% Preguntar al usuario si desea guardar el resultado en la base de datos
save = input('¿Desea guardar el resultado en la base de datos? (S/N): ', 's');

if strcmpi(save, 'S') || strcmpi(save, 's')
    % Crear tabla en base de datos si no existe
    %tablename = 'problema5';
    %query = ['CREATE TABLE ', tablename, ' (a INT, b INT, c INT, resultado VARCHAR(20));'];
    %pq_exec_params(conn, query);

    % Insertar el resultado en la tabla "problema5"
    query = ['INSERT INTO ', tablename, ' (a, b, c, resultado) VALUES (', num2str(a), ', ', num2str(b), ', ', num2str(c), ', ', "'" resultado, "'", ');'];
    pq_exec_params(conn, query);

    fprintf('Resultados guardados en la base de datos.\n');
else
    fprintf('Resultados no guardados en la base de datos.\n');
end

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 9: Código en Octave

J. Problema 10

Desarrollo un programa que calcule el factorial de un número si y solo si es divisible entre el numero 7. Si no es divisible debe mostrar un mensaje informando el error. Almacenar el resultado y opción de mostrar el historial.

```
% PROGRAMA 10 -- CÁLCULO DEL FACTORIAL SI EL NÚMERO ES DIVISIBLE POR 7
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Solicitar al usuario que ingrese un número
num = input('Ingrese un número: ');

% Verificar si el número es divisible por 7
if mod(num, 7) == 0
    fprintf('El número ingresado no es divisible por 7.\n');
else
    % Calcular el factorial del número
    fact = 1;
    for i = 1:num
        fact = fact * i;
    end

    % Mostrar el resultado
    fprintf('El factorial de %d es %d.\n', num, fact);
end

% Preguntar al usuario si desea guardar el resultado en la base de datos
save = input('¿Desea guardar el resultado en la base de datos? (S/N): ', 's');

if strcmpi(save, 'S') || strcmpi(save, 's')
    % Crear tabla en base de datos si no existe
    %tablename = 'problema10';
    %query = ['CREATE TABLE ', tablename, ' (num INT, fact BIGINT);'];
    %pq_exec_params(conn, query);
```

```
if strcmpi(save, 'S') || strcmpi(save, 's')
    % Crear tabla en base de datos si no existe
    %tablename = 'problema10';
    %query = ['CREATE TABLE ', tablename, ' (num INT, fact BIGINT);'];
    %pq_exec_params(conn, query);


    % Insertar el resultado en la tabla "problema10"
    % Convertir el valor a tipo "bigint" antes de realizar la inserción
    fact_bigint = num2str(int64(fact));
    query = ['INSERT INTO ', tablename, ' (num, fact) VALUES (', num2str(num), ', ', fact_bigint, ');'];
    pq_exec_params(conn, query);

    fprintf('Resultados guardados en la base de datos.\n');
else
    fprintf('Resultados no guardados en la base de datos.\n');
end

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 10: Código en Octave

K. Problema 11

Realizar una calculadora de áreas de figuras geométricas de: circulo, triángulo, cuadrado, rectángulo. Almacenar el resultado y opción de mostrar el historial.

```
% PROGRAMA 11 -- CALCULADORA DE ÁREAS DE FIGURAS GEOMÉTRICAS
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Crear tabla en base de datos para almacenar resultados
%tablename = 'problema11';
%query = ['CREATE TABLE ', tablename, ' (figura VARCHAR(50), a REAL, b REAL, area REAL);'];
%pq_exec_params(conn, query);

% Definir funciones para calcular áreas
function area = circulo_area(radio)
    area = pi * radio^2;
end

function area = triangulo_area(base, altura)
    area = 0.5 * base * altura;
end

function area = cuadrado_area(lado)
    area = lado^2;
end

function area = rectangulo_area(base, altura)
    area = base * altura;
end

% Menú interactivo para seleccionar figura y ingresar valores
while true
    % Solicitar al usuario que ingrese un número
    figura = input('Ingrese una figura (circulo, triangulo, cuadrado, rectangulo): ', 's');
    a = input('Ingrese el valor de a: ');
    b = input('Ingrese el valor de b(si selecciono cuadrado o circulo colocar 0): ');

    % Calcular el área de la figura seleccionada
    area = 0;
    if strcmpi(figura, 'circulo')
```

```
        area = triangulo_area(a, b);
    elseif strcmpi(figura, 'cuadrado')
        area = cuadrado_area(a);
    elseif strcmpi(figura, 'rectangulo')
        area = rectangulo_area(a, b);
    else
        fprintf('Figura no válida.\n');
        continue;
    end

    % Mostrar el resultado
    fprintf('El área de la %s es %f.\n', figura, area);

    % Almacenar el resultado en la tabla "calculadora_areas"
    query = ['INSERT INTO ', tablename, ' (figura, a, b, area) VALUES ('', figura, '', ', num2str(a), ', ', num2str(b), ', ', num2str(area), ');'];
    pq_exec_params(conn, query);

    % Preguntar al usuario si desea mostrar el historial de resultados
    show_history = input('¿Desea mostrar el historial de resultados? (S/N): ', 's');

    if strcmpi(show_history, 'S') || strcmpi(show_history, 's')
        % Consultar y mostrar el historial de resultados
        result = pq_exec_params(conn, 'select*from problema11;') %ver datos en la tabla

        if isempty(result)
            fprintf('No se encontraron resultados en el historial.\n');
        else
            fprintf('Historial de resultados:\n');
            for i = 1:size(result, 1)
                fprintf('%s: %f unidades cuadradas\n', result{i, 1}, result{i, 4});
            end
        end
    end
end

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 11: Código en Octave

L. Problema 12

Programa que recibe 3 números enteros positivos correspondientes a 3 notas, si el promedio es mayor o igual que 60 debe mostrar un mensaje de reprobado, en ambos casos debe mostrar el promedio. Almacenar el resultado y opción de mostrar el historia.

```
%PROBLEMA 12 -- PROMEDIO DE NOTAS
%Conexión con la base de datos
pkg load database
conn=pq_connect(setdbopts('dbname','cortol','host','localhost','port','5432','user','postgres','password','523811'))

% Crear la tabla si no existe
%tablename = 'problema12';
%query = ['CREATE TABLE IF NOT EXISTS ', tablename, ' (id SERIAL PRIMARY KEY, nota1 INTEGER, nota2 INTEGER, nota3 INTEGER, promedio REAL, resultado VARCHAR(10));'];
%pq_exec_params(conn, query);

% Menú interactivo para ingresar notas
while true
    % Solicitar al usuario que ingrese 3 notas
    nota1 = input('Ingrese la primera nota: ');
    nota2 = input('Ingrese la segunda nota: ');
    nota3 = input('Ingrese la tercera nota: ');

    % Calcular el promedio de las notas
    promedio = (nota1 + nota2 + nota3) / 3;

    % Determinar si el estudiante aprobó o no
    resultado = '';
    if promedio >= 60
        resultado = 'Aprobado';
    else
        resultado = 'Reprobado';
    end

    % Mostrar el resultado
    fprintf('El promedio de las notas es %f y el resultado es %s.\n', promedio, resultado);
```

```
    % Mostrar el resultado
    fprintf('El promedio de las notas es %f y el resultado es %s.\n', promedio, resultado);

    % Almacenar el resultado en la tabla
    query = ['INSERT INTO ', tablename, ' (nota1, nota2, nota3, promedio, resultado) VALUES (', num2str(nota1), ', ', num2str(nota2), ', ', num2str(nota3), ', ', num2str(promedio), ',
    pq_exec_params(conn, query);

    % Preguntar al usuario si desea mostrar el historial de resultados
    show_history = input('¿Desea mostrar el historial de resultados? (S/N): ', 's');

    if strcmpi(show_history, 'S') || strcmpi(show_history, 's')
        % Consultar y mostrar el historial de resultados
        pq_exec_params(conn,'select*from problema12;') %ver datos en la tabla
    else
        fprintf('No se encontraron resultados en el historial.\n');
    end
    break
end
% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 12: Código en Octave

M. Problema 13

Programa que recibe un numero entero positivo, correspondiente al año de nacimiento y debe mostrar si el año fue bisiesto o no. almacenar el resultado y opción de mostrar el historial.

```
%PROBLEMA 13 -- AÑO BISIESTO
%Conexión con la base de datos
pkg load database
conn = pq_connect(setdbopts('dbname', 'cortol', 'host', 'localhost', 'port', '5432', 'user', 'postgres', 'password', '523811'));

% Crear la tabla si no existe
%tablename = 'problema13';
%query = ['CREATE TABLE IF NOT EXISTS ', tablename, ' (id SERIAL PRIMARY KEY, año INTEGER, bisiesto BOOLEAN);'];
%pq_exec_params(conn, query);

% Función que verifica si un año es bisiesto
function resultado = es_bisiesto(year)
    if mod(year, 4) == 0
        if mod(year, 100) == 0
            if mod(year, 400) == 0
                resultado = true;
            else
                resultado = false;
            end
        else
            resultado = true;
        end
    else
        resultado = false;
    end
end

% Menú interactivo para ingresar el año de nacimiento
while true
    year = input('Ingrese el año de nacimiento: ');
    bisiesto = es_bisiesto(year); % Verificar si el año es bisiesto
```

```
    % Almacenar el resultado en la tabla "problema13"
    query = ['INSERT INTO ', tablename, ' (año, bisiesto) VALUES (', num2str(year), ', ''', num2str(bisiesto), ''');'];
    pq_exec_params(conn, query);

    % Mostrar el resultado
    if bisiesto
        fprintf('El año %d es bisiesto.\n', year);
    else
        fprintf('El año %d no es bisiesto.\n', year);
    end

    % Preguntar al usuario si desea ingresar otro año
    another_year = input('¿Desea verificar otro año? (S/N): ', 's');

    if ~strcmpi(another_year, 'S') && ~strcmpi(another_year, 's')
        break;
    end
end

% Preguntar al usuario si desea mostrar el historial de resultados
show_history = input('¿Desea mostrar el historial de resultados? (S/N): ', 's');

if strcmpi(show_history, 'S') || strcmpi(show_history, 's')
    % Consultar y mostrar el historial de resultados
    result = pq_exec_params(conn, {'SELECT * FROM ', tablename, ';'});
    disp(result);
end

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 13: Código en Octave

N. Problema 14

Desarrollar un programa que clasifique y almacene la información de un grupo de taxis, indicando si se encuentran en ópticas condiciones o necesitan mantenimiento si debe renovarse. Las condiciones son las siguientes: si es modelo menor a 2007 y tiene más de 20,0km recorrido debe renovarse. Si es modelo entre 2007 y 2013 tiene 20,000km debe recibir mantenimiento. Si es modelo mayor a 2013 y tiene menos de 10,000km esta en optimas condiciones. Si no cumple ninguna de las anteriores condiciones debe desplegar un mensaje que diga, "mecánico". Almacenar el resultado y opción de mostrar el historial.

```
%PROBLEMA 14 -- Verificar mantenimiento de taxis
%Conexión con la base de datos
pkg load database
conn = pq_connect(setdbopts('dbname', 'cortol', 'host', 'localhost', 'port', '5432', 'user', 'postgres', 'password', '523811'));

% Crear la tabla si no existe
%tablename = 'problema14';
%query = ['CREATE TABLE IF NOT EXISTS ', tablename, ' (id SERIAL PRIMARY KEY, modelo INTEGER, recorrido DOUBLE PRECISION, estado VARCHAR(255));'];
%pq_exec_params(conn, query);

% Función para clasificar el estado de los taxis
function estado = clasificar_taxi(modelo, recorrido)
    if modelo < 2007 && recorrido > 20000
        estado = 'Renovar';
    elseif modelo >= 2007 && modelo <= 2013 && recorrido >= 20000
        estado = 'Mantenimiento';
    elseif modelo > 2013 && recorrido < 10000
        estado = 'Óptimas condiciones';
    else
        estado = 'Mecánico';
    end
end

% Ingresar información de taxis
num_taxis = input('Ingrese el número de taxis: ');

for i = 1:num_taxis
    modelo = input('Ingrese el modelo del taxi: ');
    recorrido = input('Ingrese el recorrido del taxi (en km): ');

    % Clasificar el estado del taxi
    estado_taxi = clasificar_taxi(modelo, recorrido);
```

```
    % Almacenar la información en la tabla "problema14"
    query = ['INSERT INTO ', tablename, ' (modelo, recorrido, estado) VALUES (', num2str(modelo), ', ', num2str(recorrido), ', ''', estado_taxi, ''');'];
    pq_exec_params(conn, query);

    % Mostrar el resultado
    fprintf('El taxi con modelo %d y recorrido %f km está en estado: %s\n', modelo, recorrido, estado_taxi);
end

% Preguntar al usuario si desea mostrar el historial de resultados
show_history = input('¿Desea mostrar el historial de resultados? (S/N): ', 's');

if strcmpi(show_history, 'S') || strcmpi(show_history, 's')
    % Consultar y mostrar el historial de resultados
    result = pq_exec_params(conn, {'SELECT * FROM ', tablename, ';'});
    disp(result);
end

% Cerrar la conexión con la base de datos
pq_close(conn);
```

Figure 14: Código en Octave

III. RESULTADOS

A. Problema 1

```
>> p1_numeros

conn = <PGconn object>
Ingrese el primer número: 24
Ingrese el segundo número: 33
Ingrese el tercer número: 19
Resultado de la operación (Multiplicación): 15048
>>
```

Figure 15: Resultado de la ventana de comandos en Octave

B. Problema 2

```
>> p2_divisores

conn = <PGconn object>
Ingrese un número: 8
Los divisores de 8 son:
   1   2   4   8
>>
```

Figure 16: Resultado de la ventana de comandos en Octave

C. Problema 3

```
>> p3_vocales

conn = <PGconn object>
Ingrese una palabra: historia
La palabra historia tiene 4 vocales.
>>
```

Figure 17: Resultado de la ventana de comandos en Octave

D. Problema 4

```
>> p4_suma

conn = <PGconn object>
Ingrese un número: 7
La suma de los números desde 0 hasta 7 es: 28
>> p4_suma
conn = <PGconn object>
Ingrese un número: 28
La suma de los números desde 0 hasta 28 es: 406
>>
```

Figure 18: Resultado de la ventana de comandos en Octave

E. Problema 5

```
>> p5_dosendos

conn = <PGconn object>
Ingrese un número de inicio: 31
Ingrese un número de fin: 49
31
33
35
37
39
41
43
45
47
49
>>
```

Figure 19: Resultado de la ventana de comandos en Octave

F. Problema 6

```
>> p6_mayor
conn = <PGconn object>
Ingrese el primer número: 15
Ingrese el segundo número: 21
21
20
19
18
17
16
15
>>
```

Figure 20: Resultado de la ventana de comandos en Octave

G. Problema 7

```
>> p7_contandovocales

conn = <PGconn object>
Ingrese una palabra: restaurante
A=2, E=2, I=0, O=0, U=1
>>
```

Figure 21: Resultado de la ventana de comandos en Octave

H. Problema 8

```
>> p8_impares

conn = <PGconn object>
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57
 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
Cantidad de números impares: 50
¿Desea guardar el resultado en la base de datos? (S/N): n
Resultados no guardados en la base de datos.
>>
```

Figure 22: Resultado de la ventana de comandos en Octave

I. Problema 9

```
>> p9_triangulo

conn = <PGconn object>
Ingrese el primer lado del triángulo: 5
Ingrese el segundo lado del triángulo: 5
Ingrese el tercer lado del triángulo: 5
El triángulo es equilátero.
¿Desea guardar el resultado en la base de datos? (S/N): s
Resultados guardados en la base de datos.
>>
```

Figure 23: Resultado de la ventana de comandos en Octave

J. Problema 10

```
>> p10_factorial
conn = <PGconn object>
Ingrese un número: 21
El factorial de 21 es 5.10909e+19.
¿Desea guardar el resultado en la base de datos? (S/N): n
Resultados no guardados en la base de datos.
>>
```

Figure 24: Resultado de la ventana de comandos en Octave

K. Problema 11

```
>> p11_areas

conn = <PGconn object>
Ingrese una figura (circulo, triangulo, cuadrado, rectangulo): rectangulo
Ingrese el valor de a: 23
Ingrese el valor de b(si seleccionó cuadrado o circulo colocar 0): 9
El área de la rectangulo es 207.000000.
¿Desea mostrar el historial de resultados? (S/N): s
result =

  scalar structure containing the fields:

    data =
    {
      [1,1] = cuadrado
      [2,1] = triangulo
      [3,1] = rectangulo
      [4,1] = circulo
      [5,1] = cuadrado
      [6,1] = circulo
      [7,1] = rectangulo
      [1,2] = 10
      [2,2] = 5
      [3,2] = 12
      [4,2] = 5
      [5,2] = 5
```

Figure 25: Resultado de la ventana de comandos en Octave

L. Problema 12

```
>> p12_notas

conn = <PGconn object>
Ingrese la primera nota: 90
Ingrese la segunda nota: 23
Ingrese la tercera nota: 53
El promedio de las notas es 55.333333 y el resultado es Reprobado.
¿Desea mostrar el historial de resultados? (S/N): n
No se encontraron resultados en el historial.
>>
```

Figure 26: Resultado de la ventana de comandos en Octave

M. Problema 13

```
>> p13_bisiesto

Ingrese el año de nacimiento: 1992
El año 1992 es bisiesto.
¿Desea verificar otro año? (S/N): s
Ingrese el año de nacimiento: 1875
El año 1875 no es bisiesto.
¿Desea verificar otro año? (S/N): n
¿Desea mostrar el historial de resultados? (S/N): s
    data =
    {
      [1,1] = 1
      [2,1] = 2
      [3,1] = 3
      [4,1] = 4
      [1,2] = 2001
      [2,2] = 2008
      [3,2] = 1992
      [4,2] = 1875
      [1,3] = 0
      [2,3] = 1
      [3,3] = 1
      [4,3] = 0
    }
```

Figure 27: Resultado de la ventana de comandos en Octave

N. Problema 14

```
>> p14_taxi

Ingrese el número de taxis: 2
Ingrese el modelo del taxi: 2001
Ingrese el recorrido del taxi (en km): 10000
El taxi con modelo 2001 y recorrido 10000.000000 km está en estado: Mecánico
Ingrese el modelo del taxi: 2021
Ingrese el recorrido del taxi (en km): 5000
El taxi con modelo 2021 y recorrido 5000.000000 km está en estado: Óptimas condic
iones
¿Desea mostrar el historial de resultados? (S/N): n
>>
```

Figure 28: Resultado de la ventana de comandos en Octave

IV. BASE DE DATOS

A. Problema 1

	num1 integer	num2 integer	num3 integer	resultado text	operacion text
1	12	13	14	121314	Concatenación
2	23	12	21	56	Suma

Figure 29: Tabla en PostgreSQL con los datos ingresados en Octave

B. Problema 2

	num integer	divisores text	operacion text
1	25	1 5 25	Divisores

Figure 30: Tabla en PostgreSQL con los datos ingresados en Octave

C. Problema 3

	word text	num_vowels integer	operacion text
1	computadora	5	Vocales
2	soldado	3	Vocales

Figure 31: Tabla en PostgreSQL con los datos ingresados en Octave

D. Problema 4

	num integer	sum integer	operacion text
1	6	21	Suma

Figure 32: Tabla en PostgreSQL con los datos ingresados en Octave

E. Problema 5

	num integer	num_inicio integer	num_fin integer	operacion text
1	12	12	25	Numeros de 2 en 2
2	14	12	25	Numeros de 2 en 2
3	16	12	25	Numeros de 2 en 2
4	18	12	25	Numeros de 2 en 2
5	20	12	25	Numeros de 2 en 2
6	22	12	25	Numeros de 2 en 2
7	24	12	25	Numeros de 2 en 2

Figure 33: Tabla en PostgreSQL con los datos ingresados en Octave

F. Problema 6

	num integer	num_inicio integer	num_fin integer	operacion text
1	18	18	25	Lista de mayor a menor
2	21	21	32	Lista de mayor a menor

Figure 34: Tabla en PostgreSQL con los datos ingresados en Octave

G. Problema 7

	palabra text	cont_a integer	cont_e integer	cont_i integer	cont_o integer	cont_u integer
1	silla	1	0	1	0	0
2	matematica	3	1	1	0	0

Figure 35: Tabla en PostgreSQL con los datos ingresados en Octave

H. Problema 8

	numero integer	opcion character varying (20)
1	1	impares
2	3	impares
3	5	impares
4	7	impares
5	9	impares
6	11	impares
7	13	impares
8	15	impares
9	17	impares
10	19	impares
11	21	impares
12	23	impares
13	25	impares
14	27	impares
15	29	impares
16	31	impares
17	33	impares

	numero integer	opcion character varying (20)
35	69	impares
36	71	impares
37	73	impares
38	75	impares
39	77	impares
40	79	impares
41	81	impares
42	83	impares
43	85	impares
44	87	impares
45	89	impares
46	91	impares
47	93	impares
48	95	impares
49	97	impares
50	99	impares
51	50	cantidad

Figure 36: Tabla en PostgreSQL con los datos ingresados en Octave

I. Problema 9

	a integer	b integer	c integer	resultado character varying (20)
1	6	6	4	resultado
2	6	10	8	escaleno
3	5	5	5	equilátero
4	11	11	9	isósceles

Figure 37: Tabla en PostgreSQL con los datos ingresados en Octave

J. Problema 10

	num integer	fact bigint
1	7	5040
2	14	87178291200

Figure 38: Tabla en PostgreSQL con los datos ingresados en Octave

K. Problema 11

	figura character varying (50)	a real	b real	area real
1	cuadrado	10	20	100
2	triangulo	5	6	15
3	rectangulo	12	16	192
4	circulo	5	2	78.5398
5	cuadrado	5	5	25
6	circulo	8	0	201.062

Figure 39: Tabla en PostgreSQL con los datos ingresados en Octave

L. Problema 12

	id [PK] integer	nota1 integer	nota2 integer	nota3 integer	promedio real	resultado character varying (10)
1	1	94	56	88	79.3333	Aprobado
2	2	45	34	72	50.3333	Reprobado
3	3	23	99	66	62.6667	Aprobado
4	4	32	45	67	48	Reprobado

Figure 40: Tabla en PostgreSQL con los datos ingresados en Octave

M. Problema 13

	id [PK] integer	año integer	bisiesto boolean
1	1	2001	false
2	2	2008	true

Figure 41: Tabla en PostgreSQL con los datos ingresados en Octave

N. Problema 14

	id [PK] integer	modelo integer	recorrido double precision	estado character varying (255)
1	1	2007	21000	Mantenimiento
2	2	2014	10000	Mecánico
3	3	2018	25000	Mecánico

Figure 42: Tabla en PostgreSQL con los datos ingresados en Octave

V. GITHUB

https://github.com/ac428/Corto1-proyectos-IE