1. Diseñar un algoritmo en pseudocódigo que:
    a. Cargue un arreglo de caracteres.
    b. Permita al usuario elegir si lo quiere ver en el orden ingresado o invertido (Modularice apropiadamente)

2. Diseñar un algoritmo en pseudocódigo que lea un valor entero (N) y genere un arreglo con los 10 primeros múltiplos del mismo. Por ejemplo para N=7 deberá guardar en el arreglo: 7 14 21 28 35 42 49 56 63 70

3. Diseñar un algoritmo en pseudocódigo que dado un valor entero N y un arreglo de enteros, reemplace los valores en las posiciones pares del arreglo por el valor N y muestre el arreglo resultante.

4. Diseñar un algoritmo en pseudocódigo que permita encontrar el valor más grande y el más pequeño almacenado en un arreglo de números.

5. Diseñar un algoritmo en pseudocódigo que calcule el promedio de los valores almacenados en un arreglo de números.

6. Diseñar un algoritmo en pseudocódigo que permita almacenar letras en un arreglo, cuya dimensión máxima es de 100 posiciones. El algoritmo debe verificar que el caracter leído sea una letra antes de guardarlo en el arreglo. Al finalizar la carga el algoritmo debe mostrar por pantalla la cantidad de letras guardadas.

7. Diseñar un algoritmo en pseudocódigo que permita: a. Leer palabras y almacenarlas en un arreglo de string. b. Generar una cadena con las palabras almacenadas en el arreglo separándolas por un espacio en blanco c. Generar otra cadena con las palabras almacenadas en el arreglo en orden inverso separándolas por un guión ('-') d. Mostrar ambas cadenas por pantalla.

8. Diseñar un algoritmo en pseudocódigo que busque la palabra más larga almacenada en un arreglo de String (cada posición guarda exactamente 1 palabra).

9. Diseñar dos módulos en pseudocódigo que dado un arreglo de caracteres y un caracter: a. Verifique si el caracter ingresado se encuentra en el arreglo. ¿Puede optimizar el algoritmo? b. Cuente cuántas veces aparece el caracter en el arreglo. ¿Puede optimizar el algoritmo? Implementar el algoritmo llamador que invoque a los módulos.

10. Diseñar un algoritmo en pseudocódigo que dado un arreglo cargado con valores fijos genere otro arreglo con los valores invertidos. Por ejemplo si el arreglo contiene: 12 4 8 22 5, el nuevo arreglo será 5 22 8 4 12

11. Diseñar un algoritmo en pseudocódigo que cargue dos arreglos de números y luego verifique si son iguales o no. Para ello se debe implementar un módulo que realice la verificación.

12. Diseñar un algoritmo en pseudocódigo que cargue un arreglo de caracteres y luego realice la copia de un arreglo en otro de igual tamaño (modularice). 13. Diseñar un algoritmo en pseudocódigo que cargue un arreglo de caracteres y luego genere otro que contenga solo las vocales que se encuentran en el arreglo original.

14. Diseñar un algoritmo en pseudocódigo que cargue un arreglo de String y luego genere dos nuevos arreglos, uno conteniendo las cadenas que estaban en las posiciones pares y otro conteniendo los caracteres que estaban en las posiciones impares. Modularice.

```
ALGORITMO arregloParImpar() RETORNA ∅
(*Este algoritmo genera 2 nuevos arreglos mostrando los caracteres que hay en la posiciones par e impares de un arreglo padre.*)

ENTERO longitudArreglo ← 0, longitudPar, longitudImpar
longitudArreglo ← verificarLongitudArreglo()
longitudImpar ←numerosImpar(longitudArreglo)
longitudPar←(longitudArreglo-longitudImpare)
TEXTO [] almacenTexto←CREAR TEXTO [longitudArreglo]
cargarTexto(almacenTexto)
TEXTO [] almacenImpares←CREAR TEXTO[longitudImpar]
cargarImpares(almacenTexto)
TEXTO [] almacenPares←CREAR TEXTO[longitudPar]
cargarPares(almacenTexto)
mostrarArreglo(almacenTexto,almacenImpares, almacenPares)
```

```
MODULO verificarLongitudArreglo() RETORNA ENTERO
(*Verifica si el número ingresado por el usuario es un número válido para la longitud de arreglo*)
        ENTERO longitud ← 0
        REPETIR
                ESCRIBIR (“¿Cuántos números desea ingresar?”)
                LEER (longitud)
                SI longitud <= 0 ENTONCES
                        ESCRIBIR (“El número ingresado es incorrecto. Por favor, inténtelo nuevamente”)
                FIN SI
        HASTA (longitud <= 0)
        RETORNA longitud
FIN MODULO

MODULO cargarTexto(TEXTO [] almacenTexto) RETORNA ∅
(*Este módulo se encarga de cargar valores arreglo.*)
        ENTERO i, logitudTexto
        STRING textoTemporal← “ ”
        longitudTexto←LONGITUD(almacenTexto)-1
        PARA i←0, HASTA longitudTexto-1, PASO 1, HACER
                ESCRIBIR(“Ingrese el ” + (i + 1) +“° texto: ”)
                LEER (textoTemporal)
                almacenTexto[ i ] ← textoTemporal
        FIN PARA
FIN MÓDULO

MODULO numeroImpar (ENTERO longitudArreglo) RETORNA ENTERO
(*Este modulo se encarga de saber cuantos numeros pares tiene un numero padre*)
        ENTERO i, numeroPar←0
                PARA i←0 HASTA longitudArreglo, PASO 1 HACER
                numeroPar←longitudArreglo DIV 2
        FIN PARA
RETORNA numeroPar
FIN MODULO

MODULO cargarImpares (TEXTO [] almacenTexto, TEXTO [] almacenImpares) RETORNA ∅
(*Este modulo se encargar de cargar el arreglo impar*)
        ENTERO i, longitudTexto
        longitudTexto←LONGITUD(almacenImpares)-1
                PARA i←0 HASTA longitudTexto, PASO 1 HACER
                almacenImpares[i]=almacenTexto[(i*2)+1]
        FIN PARA
FIN MODULO

MODULO cargarPares (TEXTO [] almacenTexto, TEXTO [] almacenPar) RETORNA ∅
(*Este modulo se encargar de cargar el arreglo par*)
        ENTERO i, longitudTexto
        longitudTexto←LONGITUD(almacenPar)-1
                PARA i←0 HASTA longitudTexto, PASO 1 HACER
                almacenPares[i]=almacenTexto[(i*2)]
        FIN PARA
FIN MODULO

MODULO mostrarArreglo(TEXTO [] almacenTexto, TEXTO [] almacenPares, TEXTO [] almacenImpares) RETORNA ∅
(*Este modulo se encargar de mostrar por pantalla los 3 arreglos*)
        ENTERO i, longitudTexto1, longitudTexto2, longitudTexto3
        longitudTexto1←LONGITUD(almacenTexto)
        longitudTexto2←LONGITUD(almacenImpares)
        longitudTexto3←LONGITUD(almacenPares)
        ESCRIBIR(“Los datos del arreglo padre son:”)
                PARA i←0 HASTA longitudTexto1, PASO 1 HACER
                        ESCRIBIR(almacenTexto[i]+ “ ”)
        FIN PARA
        ESCRIBIR(“\nLos datos del las posiciones impares son:”)
                PARA  i←0 HASTA longitudTexto2, PASO 1 HACER
                        ESCRIBIR(almacenImpares[i]+ “ ”)
        FIN PARA
        ESCRIBIR(“\nLos datos del las posiciones pares son:”)
                PARA  i←0 HASTA longitudTexto3, PASO 1 HACER
                        ESCRIBIR(almacenPares[i]+ “ ”)
        FIN PARA
FIN MODULO
```

15. Problema Número de DNI. El documento de identidad (DNI) en España, consta de 8 cifras y de una letra. La letra del DNI se obtiene siguiendo los pasos a continuación:
1) Calcula el resto de dividir el número del DNI entre 23
2) El número obtenido estará entre 0 y 22, selecciona la letra asociada al valor obtenido utilizando la siguiente tabla:

Por ejemplo, si el número del DNI es 31415927 y el resto de dividir por 23 es 20, la letra que le corresponde según la tabla es la "C" Diseñar un algoritmo que solicite un numero de 8 cifras y devuelva el número de DNI correspondiente.

16. Dado un arreglo que almacena cadenas de caracteres se desea verificar que las mismas cumplan con las siguientes condiciones: tengan una longitud mínima de 5 caracteres y que contenga solo letras. En caso de que la cadena no cumpla la condición debe ser eliminada del arreglo y la cadena que está en la siguiente posición debe ocupar su lugar. Imprima por pantalla el arreglo resultante.

```
ENTERO longitudArreglo ← 0
longitudArreglo ← verificarLongitudArreglo()
TEXTO [] almacenTexto←CREAR TEXTO [longitudArreglo]
TEXTO [] almacenTexto2←CREAR TEXTO [longitudArreglo]
cargarTexto(almacenTexto)
verificarArreglo(almacenTexto, almacenTexto2)
modificarArreglo(almacenTexto2)
mostrarArreglo(almacenTexto2)
```

```
MODULO verificarLongitudArreglo() RETORNA ENTERO
(*Verifica si el número ingresado por el usuario es un número válido para la longitud de arreglo*)
        ENTERO longitud ← 0
        REPETIR
                ESCRIBIR ("¿Cuántos números desea ingresar?")
                LEER (longitud)
                SI longitud <= 0 ENTONCES
                        ESCRIBIR ("El número ingresado es incorrecto. Por favor, inténtelo nuevamente")
                FIN SI
        HASTA (longitud <= 0)
        RETORNA longitud
FIN MODULO

MODULO cargarTexto(TEXTO [] almacenTexto) RETORNA ∅
(*Este módulo se encarga de cargar valores arreglo.*)
        ENTERO i, logitudTexto
        STRING textoTemporal← " "
        longitudTexto←LONGITUD(almacenTexto)-1
        PARA i←0, HASTA longitudTexto-1, PASO 1, HACER
                ESCRIBIR("Ingrese el " + (i + 1) +"° texto: ")
                LEER (textoTemporal)
        FIN PARA
FIN MÓDULO

MODULO identificarLetras(TEXTO cadena) RETORNA LOGICO
(*Este modulo indentifica si la palabra esta formada por letras*)
        ENTERO i←0 , largoCadena
        LOGICO esLetra←VERDADERO
        largoCadena←LONGITUD(cadena)
        REPETIR
                SI !Character.isLetter(cadena.charAt(i) ENTONCES
                        esLetra←FALSO
                FIN SI
                i++
        HASTA esLetra AND i<largoCadena
RETORNA esLetra
FIN MODULO
```

```
MODULO verificarArreglo(TEXTO [] almacenTexto, TEXTO [] almacenTexto2) RETORNA ∅
(*Verifica si las palabras ingresadas tienen 5 caracteres de solo letras.*)
        ENTERO i, logitudTexto, j=0;
        LOGICO letra
        longitudTexto←LONGITUD(almacenTexto)-1
        PARA i←0, HASTA longitudTexto-1, PASO 1, HACER
                letra←identificarLetras(almacenTexto[i])
                SI LONGITUD(almacenTexto[i]) = 5 AND letra=VERDADERO ENTONCES
                        almacenTexto2[j] ← almacenTexto[i]
                        j++
                FIN SI
        FIN PARA
FIN MÓDULO

MODULO modificarArreglo(TEXTO [] almacenTexto2) RETORNA ∅
(*Hace que las celdas que estan en null queden en vacio para que no se vea por pantalla.*)
        ENTERO i, logitudTexto
        longitudTexto←LONGITUD(almacenTexto2)-1
        PARA i←0, HASTA longitudTexto-1, PASO 1, HACER
                SI almacenTexto2[i] = null ENTONCES
                        almacenTexto2[i] ← “ ”
                FIN SI
        FIN PARA
FIN MÓDULO

MODULO mostrarArreglo(TEXTO [] almacenTexto2) RETORNA ∅
(*Este modulo se encargar de mostrar por pantalla el arreglo.*)
        ENTERO i, logitudTexto
        longitudTexto←LONGITUD(almacenTexto2)-1
        ESCRIBIR(“\n”)
        ESCRIBIR("Los datos del arreglo son")
        PARA i←0, HASTA longitudTexto-1, PASO 1, HACER
                ESCRIBIR(almacenTexto2[i]+ “ ”)
        FIN PARA
                ESCRIBIR(“\n”)
FIN MÓDULO
```

17. Dado un arreglo que almacena las notas correspondientes a un alumno, las cuales son números reales, se desea verificar si el alumno aprobó el cuatrimestre. La condición para aprobar es tener todas las notas con valores mayores o iguales a 6. Se debe implementar un algoritmo que cargue el arreglo con 10 notas y verifique si el alumno aprobó o no el cuatrimestre.

18. Implementar un algoritmo que utilice dos arreglos, uno que almacena nombres de personas empleadas en una empresa y otro que almacena los sueldos de las mismas, y sabiendo que ambos arreglos se corresponden por posición, presentar un menú de opciones para realizar algunas de las siguientes acciones: a. Buscar la persona que tiene mayor sueldo, mostrar su nombre y el sueldo. b. Listar todas las personas que cobran exactamente un valor X (leído por teclado). c. Aumentar en un 10% los sueldos que sean inferiores a $10000. d. Buscar una persona y si se encuentra mostrar su sueldo