**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)**

```
ALGORITMO invierteOrden RETORNA ∅
(*Carga un arreglo de caracteres y muestra en orden ingresado O invertido*)
        ENTERO longitudArreglo
        CARACTER respuesta← n
        longitudArreglo ← verificarLongitudArreglo()
        CARACTER [] almacenCaracter ←CREAR CARACTER [longitudArreglo]
        cargarArreglo (almacenCaracter)

FIN ALGORITMO

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 caracteres 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 cargarArreglo (CARACTER[] almacenCaracter) RETORNA ∅
(*Este módulo se encarga de cargar valores numéricos enteros a un arreglo.*)
        ENTERO i,
        CARACTER caracter← “”
        PARA i←0, HASTA LONGITUD(almacenCaracter)-1, PASO 1, HACER
                ESCRIBIR(“Ingrese el ” + (i + 1) +“° carácter: ”)
                LEER (caracter)
                almacenCaracter[ i ] ← caracter
        FIN PARA
FIN MÓDULO

MODULO monstrarArregloInvertido (CARACTER[] almacenCaracter) RETORNA ∅
(*Este modulo se encarga de invertir la visualización del arreglo*)
        ENTERO i,
        CARACTER caracter← “”
        PARA i←0, HASTA LONGITUD(almacenCaracter)-1, PASO 1, HACER
                caracteres ← almacenCaracter [i] + caracteres
        FIN PARA
FIN MÓDULO

MODULO mostrarArreglo (CARACTER[] almacenCaracter) RETORNA ∅
(*Este modulo se encarga de invertir la visualización del arreglo*)
        ENTERO i,
        TEXTO caracteres← “”
        PARA i←0, HASTA LONGITUD(almacenCaracter)-1, PASO 1, HACER
                caracteres ← caracteres + almacenCaracter [i]
        FIN PARA
FIN MÓDULO
```

**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.**

```
ALGORITMO contenerSoloVocales RETORNA ∅
(*Cargado un arreglo de caracteres, genera un nuevo arreglo con vocales.*)
        ENTERO longitudArreglo ← 0
        longitudArreglo ← verificarLongitudArreglo()
        CARACTER [] almacenCaracter ←CREAR CARACTER [longitudArreglo]
        cargarArreglo(almacenCaracter)
        SI cuentaVocales(almacenCaracter) == 0 ENTONCES
                ESCRIBIR (“El arreglo no tiene vocales.”)
        SINO
                ESCRIBIR("El nuevo modulo de vocales es: \n ")
                leerArreglo(generarArregloVocales(almacenCaracter))
        FIN SI
        generarArregloVocales (CARACTER[] almacenCaracter)

FIN ALGORITMO

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 caracteres 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 cargarArreglo (CARACTER[] almacenCaracter) RETORNA ∅
(*Este módulo se encarga de cargar valores de tipo caracter a un arreglo.*)
```

```
        ENTERO i, longitudAlmacen
        CARACTER caracter← “”
        longitudAlmacen ← LONGITUD(almacenCaracter)-1,
        PARA i←0, HASTA longitudAlmacen PASO 1, HACER
                ESCRIBIR(“Ingrese el ” + (i + 1) +“° carácter: ”)
                LEER (caracter)
                almacenCaracter[ i ] ← caracter
        FIN PARA
FIN MÓDULO

MODULO cuentaVocales (CARACTER[] almacenCaracter) RETORNA ENTERO
(*Cuenta cuántas vocales hay en un arreglo. *)
        ENTERO i, contador ← 0, longitudAlmacen
        longitudAlmacen ← LONGITUD (almacenCaracter)

        PARA i← 0 HASTA longitudAlmacen-1 PASO 1 HACER
                SI (almacenCaracter[i] == ‘a’|| almacenCaracter[i] == ‘e’ || almacenCaracter[i] == ‘i’
                || almacenCaracter[i] == ‘o’ || almacenCaracter[i] == ‘u’ )ENTONCES
                        contador ←contador +1
                FIN SI
      FIN PARA
      RETORNA contador;
FIN MODULO


MODULO generarArregloVocales (CARACTER[] almacenCaracter) RETORNA CARACTER[]
(*Este módulo se encarga de crear un nuevo arreglo con las vocales del arreglo principal. *)
        ENTERO i, longitudAlmacenCaracter, longitudAlmacenVocales, contadorDeLongitud ←0
        longitudAlmacenCaracter ← LONGITUD(almacenCaracter)
        longitudAlmacenVocales ← cuentaVocales (almacenCaracter)
        CARACTER [] almacenVocales ←CREAR CARACTER [longitudAlmacenVocales]

        PARA i←0, HASTA longitudAlmacenCaracter PASO 1, HACER
                SI (almacenCaracter[i] == ‘a’|| almacenCaracter[i] == ‘e’ || almacenCaracter[i] == ‘i’
                || almacenCaracter[i] == ‘o’ || almacenCaracter[i] == ‘u’) ENTONCES
                        almacenVocales [contadorDeLongitud]←almacenCaracter[i]
                        contadorDeLongitud ← contadorDeLongitud +1
                FIN SI
        FIN PARA
        RETORNA almacenVocales
FIN MÓDULO

MODULO leerArreglo(CARACTER[] arreglo) RETORNA ∅
(*Muestra por pantalla los caracteres de un arreglo*)
        ENTERO i, longitudArreglo
        longitudArreglo← LONGITUD(arreglo)
        ESCRIBIR (“Las vocales del arreglo original son: ”)
        PARA i←0 HASTA longitudArreglo-1 PASO 1 HACER
                ESCRIBIR (arreglo[i])
        FIN PARA
FIN MODULO
```

**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.**

**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:**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| T | R | W | A | G | M | Y | F | P | D | X | B |

| 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N | J | Z | S | Q | V | H | L | C | K | E | |

**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.**

```
ALGORITMO dniEspania RETORNA ∅
(*Este algoritmo muestra por pantalla el documento español con su letra correspondiente.*)
        ENTERO numeroDocumento
        CARACTER letraDocumento
        numeroDocumento ← verificarIngreso()
        letraDocumento ← asignaLetra(numeroDocumento)
        ESCRIBIR ("El número de documento completo es: "+ numeroDocumento + " " +
                        letraDocumento)
FIN ALGORITMO

MODULO verificaIngreso() RETORNA ENTERO
(*Verifica si el número de documento ingresado es correcto. Retorna el número entero ingresado.*)
        ENTERO numeroDocumento
        REPETIR
                ESCRIBIR ("Ingrese el número de documento: ")
                LEER (numeroDocumento)
                SI (numeroDocumento < 10 000 000 OR numeroDocumento >= 100 000 000)ENTONCES
                        ESCRIBIR ("El número ingresado es incorrecto. Por favor, inténtelo nuevamente")
                FIN SI
        HASTA (numeroDocumento < 10 000 000 OR numeroDocumento >= 100 000 000)
        RETORNA numeroDocumento
FIN MODULO

MODULO asignaLetra(ENTERO documento) RETORNA CARACTER
(*Dado un número de documento, calcula el resto de la división por 23 del documento y retorna la letra correspondiente.*)
        ENTERO resto
        CARACTER letraAsignada ←""
        CARACTER[] almacenLetras ← { 'T', 'R', 'W', 'A', 'G', 'M', 'Y', 'F', 'P', 'D', 'X', 'B', 'N', 'J', 'Z', 'S', 'Q', 'V', 'H', 'L', 'C', 'K', 'E'}
        resto ← documento MOD 23
        letraAsignada ← almacenLetras[resto]
        RETORNA letraAsignada
FIN MODULO
```

**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.**

**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 (leítdo 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**

```
ALGORITMO empleadosSueldos RETORNA ∅
(*Asumiendo existen dos arreglos existentes de empleados y sueldos realiza diferentes acciones sobre ellos.*)
        CARACTER opcionUsuario
        ENTERO valorSueldo
        TEXTO nombreCompletoEmpleado
        TEXTO[] empleados ← {"Leo Bruno", "Jeremias Herrera", "Maria Monserrat", "Axl Vidman"}
```

```
        REAL[] sueldos ← {12000, 9000, 10000, 10000}
        opcionUsuario ← menu()

        SEGUN (opcionUsuario) HACER
                CASO ‘a’:
                        ESCRIBIR ("El mayor sueldo es de $" + sueldos[sueldoMayor(sueldos)] + " y le pertenece a " +
                        empleados[sueldoMayor(sueldos)] + ". ")
                CASO ‘b’:
                        ESCRIBIR("Ingrese un valor (sueldo) que desee buscar: ")
                        LEER (valorSueldo)
                        sueldosIguales(sueldos, empleados, valorSueldo)
                CASO ‘c’:
                        aumentarSueldos(sueldos, empleados)
                CASO ‘d’:
                        ESCRIBIR ("Ingrese el nombre y apellido del empleado: ")
                        LEER (nombreCompletoEmpleado)
                        buscaUsuario(sueldos, empleados, nombreCompletoEmpleado)
                PREDETERMINADO
                        ESCRIBIR (“Programa cerrado.”)
        FIN SEGUN
FIN ALGORITMO

MODULO sueldoMayor(REAL[] sueldo) RETORNA ENTERO
(*Busca en el arreglo de sueldos el mayor sueldo y devuelve la ubicación en el arreglo.*)
        ENTERO i, posicion ← 0, longitudArreglo
        REAL sueldoMayor ← 0
        longitudArreglo ← LONGITUD(sueldo)
        PARA i ← 0 HASTA longitudArreglo-1 PASO 1 HACER
                SI (sueldo[i] > sueldoMayor) ENTONCES
                        sueldoMayor ← sueldo[i]
                        posicion ← i
                FIN SI
        FIN PARA
        RETORNA posicion
FIN MODULO

MODULO sueldosIguales(REAL[] sueldos, TEXTO[] empleados, REAL valorSueldo) RETORNA ∅
(*Dado un valor por parametro muestra por pantalla los empleados que tienen el mismo sueldo.*)
        ENTERO i, longitudArreglo
        LOGICO noHaySueldoIgual ← VERDADERO
        longitudArreglo ← LONGITUD(sueldos)
        PARA I← 0 HASTA longitudArreglo-1 PASO 1 HACER
                SI (sueldos[i] = valorSueldo) ENTONCES
                        ESCRIBIR ("Empleado/a: " + empleados[i] + ". \n")
                        noHaySueldoIgual ← FALSO
                FIN SI
        FIN PARA

        SI (noHaySueldoIgual) ENTONCES
                ESCRIBIR ("No hay empleados con el sueldo igual al ingresado. ")
        FIN SI
FIN MODULO

MODULO aumentarSueldos(REAL[] sueldos, TEXTO[] empleados) RETORNA ∅
(*Aumenta automaticamente los sueldos inferiores a 10.000 en un 10%. Informa a quienes se les aumentó y a cuanto.*)
        ENTERO i, longitudArreglo
        LOGICO sueldoSuperiores ← VERDADERO
        longitudArreglo ← LONGITUD(sueldos)
        PARA i ← 0 HASTA longitudArreglo -1 PASO 1 HACER
                SI (sueldos[i] < 10000) ENTONCES
                        sueldos[i] = sueldos[i] + ((sueldos[i] * 10) / 100)
                        ESCRIBIR ("Se aumentó el sueldo de " + empleados[i] + " a $" + sueldos[i] + ".\n")
                        sueldoSuperiores ← FALSO
                FIN SI
        FIN PARA
        SI (sueldoSuperiores) ENTONCES
```

```
            ESCRIBIR("Ningún empleado tiene un sueldo inferior a $10000.");
        FIN SI
FIN MODULO

MODULO buscaUsuario(REAL[] sueldos, TEXTO[] empleados, TEXTO empleadoNombre) RETORNA ∅
(*Dado un nombre por parámetro busca el empleado y muestra nombre y sueldo.*)
        ENTERO i ← 0, longitudArreglo
        LOGICO empleadoNoEncontrado ← VERDADERO
        longitudArreglo ← LONGITUD(sueldos)
        HACER
            SI(empleadoNombre.equalsIgnoreCase(empleados[i])) ENTONCES
                ESCRIBIR("El empleado " + empleados[i] + " tiene un sueldo de $" + sueldos[i] + ". ")
                empleadoNoEncontrado ← FALSO
            FIN SI
            i← i+1
        MIENTRAS (empleadoNoEncontrado AND i < longitudArreglo)
        SI (empleadoNoEncontrado) ENTONCES
            ESCRIBIR ("El nombre y apellido ingresado no pertenece a ningún empleado cargado.")
        FIN SI
FIN MODULO
```