Ejercicios Complementarios

Estos ejercicios son ejercicios propuestos o de modelos de anexo a los dado en la guía o en clases. **NO** quiere decir que se tomarán estos mismos, sólo son de práctica para el alumno.

1)Busque un elemento en un arreglo y retorne la posición. Lo debe realizar utilizando el **concepto de bloque** con dos parámetros de entrada

2)Dado un texto, determinar la cantidad de vocales participantes, totalizando solamente las que participan.

3)Obtener la cantidad de los primeros N números múltiplos de M

4)Dada una lista de números, previamente ingresadas por el usuario, determinar el menor y mayor número de la lista.

5)Limpiar los caracteres con acentos en una frase y devolverla sin ellos. Ejemplo: 'hólá mÚndÓ' -> 'hola mUndO'

6)Dado un array con 30 enteros, devolver otro array's con los números que se encuentren en posiciones impares, elevados a la cuarta potencia. Nota: considerar que disponemos ya del arreglo cargado con los valores.

7)Conociendo la Serie de Fibonacci, devolver el n-esimo elemento.

8)Dado un texto, devolver en forma de colección ordenada según el tamaño de la palabra.

9)Dada una lista de enteros, devolver una lista donde el primer elemento sea la suma en absoluto de los negativos. El resto de la lista se compone por los elementos positivos de la lista original.

10)Dado un arreglo, obtener otro eliminando el valor solicitado

11)Realizar la carga de enteros positivos sin que se produzcan duplicaciones y en una estructura del tipo array. Cantidad flexible al usuario. Se debe devolver un array.

11)Dado una frase, devolver las palabras que tengan cantidad par de vocales

12)Dada una frase inicial, un parámetro a cambiar y otro por el cual se quiere cambiar, obtener la frase final con dicho cambio. Ej: 'Hola Mundo Cruel' cambiar:'u' por:'x' ->'Hola Mxndo Crxel'

13)Diseñar un algoritmo que permita encontrar un grupo de "NÚMEROS SOCIABLES". Los números sociables, cumplen con la misma condición que los números amigos, pero en vez

de ir en parejas van en grupos más grandes. La suma de los divisores del primer número da el segundo, la suma de los del segundo da el tercero, y así sucesivamente. La suma de divisores del último da el primer número de la lista. Por ejemplo los números 12496,14288,15472,14536,14264

14)Dada una cadena con letras y números, obtener otra cadena que solo tenga letras. Ej: 'Las1 clase22s 5practicas de p3aradig55mas se v1an 24 complicando'-> 'Las clases practicas de paradigmas se van complicando'

15)Realizar un programa, tal que dada una lista de cadenas, y otra lista de enteros, devuelva la lista de aquellas cadenas cuya longitud coincide con el entero correspondiente en la segunda lista.
Ej.: consistentes ['yo', 'no','te', 'digo', 'que', 'no'] [2,5,3,7] . ['yo']

16)Dada una frase inicial, quitar las palabras repetidas.

17)Un número es primo perfecto si y solo si cumple estas 3 condiciones:

1- El número es primo.
2- Cada dígito del número es primo.
3- La suma de todos los dígitos es un número primo.
EJ: 227 -> 227 (es primo), 2 es primo , 7 es primo y la suma de 2 2 y 7 es 11, y este también es primo.
Determinar si un número ingresado por el usuario es primo perfecto.

18)Dada una frase y un substrings(no precisamente una palabra, puede ser una palabra incompleta), obtener la posición donde inicia dicho substrings. En caso de no encontrar devolver 0 y de haber más de dos solo tomar el primero.

19)Dado un string de entrada y un substring, quitar del string original el substring en cuestión. Ej. 'Necesito practicar mas Smalltalk sino no llego a los parciales', quitar: 'si'-> 'Neceto practicar mas Smalltalk no no llego a los parciales'

20)Dado un texto, interpretar las señales de escape y decodificarlo.La señal de escape està dada por "~", seguido del caracter y ocurrencias del mismo. EJ: 'Hola M~u2ndo'='Hola Muundo' y hacer el proceso inverso.

21)Dada una matriz cuadrada, de nxn ingresada por el usuario, devolver otra rotada a 90º.
Nota: la matriz de entrada y salida usa un arreglo unidimensional. Ej:

Matriz original:	Matriz rotada:
1 2 3	7 4 1
4 5 6	8 5 2
7 8 9	9 6 3

22)Definir un método tal que dado un texto y una lista de caracteres, retorne el texto depurado, sin las repeticiones consecutivas de los caracteres dados.
Ej.: depura 'hola,,, todo bien,,, y uds..?????' [',,' ',.','?'] Devuelve: 'hola, todo bien, y uds.?'

23)Clave MURCIELAGO. Esta clave tiene la diferencia de que cada letra que conforma la clave es reemplazada por un número. Mira el ejemplo:

M	**U**	**R**	**C**	**I**	**E**	**L**	**A**	**G**	**O**
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

La palabra PARADIGMAS quedaría codificada como: **P727D4807S**.
Realizar la codificación de una frase dada y viceversa (decodificación).

24)Dada una matriz de nxm, donde n son las cantidades de filas por m cantidades de columnas, obtener un listado de las coordenadas de todas las coincidencias según un valor dado. La matriz se encuentra representada en una estructura unidimensional (en un arreglo, no arreglo de arreglos). EJ: Dada una matriz como:

fila/columna	columna 1	columna 2	columna 3	columna 4
fila 1	2	-1	0	'hola'
fila 2	8	4	2	$a
fila 3	1	2	34	51

se encuentra representada en un arreglo como: (2 -1 0 'hola' 8 4 2 $a 1 2 34 51)
obtener coincidencias de: 2 resultado-> ((1,1)(,2,3)(3,2))
Nota: hacer uso de la fórmula matemática para dada la fila y columna poder obtener la posición en el vector.

25)Conjetura del Número capicúa. Esta conjetura clásica se basa en: tomar un número entero positivo cualquiera. El número se escribe entonces en orden inverso; los dos números se suman. El proceso se repite con el número sumado, obteniéndose entonces una segunda suma, y se prosigue de igual forma hasta lograr un capicúa. La conjetura afirma que tras número finito de adiciones terminará por obtenerse un capicúa. EJ:

N = 42	N = 28	N = 87
42 + 24 = 66	28 + 82 = 110	87 + 78 = 165
	110 + 011 = 121	165 + 561 = 726
		726 + 627 = 1353
		1353 + 3531 = 4884

Dado un número ingresado por el usuario, determinar su número capicúa según la conjetura e indicar en cuántas iteraciones se realizó.

26)Distancia de Levenshtein. La **distancia de Levenshtein**, **distancia de edición** o **distancia entre palabras** es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra.

Por ejemplo, la distancia de Levenshtein entre "casa" y "calle" es de 3 porque se necesitan al menos tres ediciones elementales para cambiar uno en el otro.

1. casa → cala (sustitución de 's' por 'l')
2. cala → calla (inserción de 'l' entre 'l' y 'a')
3. calla → calle (sustitución de 'a' por 'e')

a)Desarrollar un programa que dado dos palabras devuelva la longitud de levenshtein entre ellas.

b)Desarrollar un programa que dado dos frases, devuelve la longitud de levenshtein en total entre ellas.

27)Resolver la SERIE: $\sum_{i=1}^{n} (x \times a^{\,i}) \div (2 \times i)$

La condición de fin debe ser llegar al último elemento (a_n) o que la diferencia entre los dos últimos elementos calculados sea menor que un número ε que el usuario ingrese.

Esto debe devolver la suma de todos los elementos calculados.

28)Hacer la productoria: $\prod_{i=1}^{1000} (2x + 3)^{i}$, detenerse cuando n=1000 o bien cuando: $|(2x+3)^n - (2x+3)^{n-1}| < e$. El x y e son ingresados por el usuario.

29)Dada la siguiente fórmula de Leibniz para el cálculo de π= 4*($\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$), obtener el valor de π, dado un n y un ϵde control que responde a: $|\sum \frac{(-1)^{n-1}}{2(n-1)+1} \sum \frac{(-1)^n}{2n+1}| < \epsilon$.

30)Dada la siguiente fórmula de Wallis para el cálculo de π/2= $\prod_{n=1}^{\infty} (\frac{2n}{2n-1} * \frac{2n}{2n+1})$, obtener el valor de π, dado un n y un ϵde control que verifique la diferencia entre los dos últimos elementos calculados sea menor que el mismo.

31)Calcular el número e, teniendo en cuenta el siguiente límite:

$$e = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n.$$

Épsilon: se debe definir un valor entero positivo, que controle la diferencias entre un término y el anterior. Terminar cuando esa diferencia es despreciable (ínfima).

n: cantidad de veces a iterar. Básicamente se lo usa para control en el caso de que las diferencias no se cumplan (punto anterior).

32)Calcular el número e, teniendo en cuenta la siguiente sumatoria:

$$e = \frac{1}{2} \sum_{k=0}^{\infty} \frac{k+1}{k!}$$

Dado: Épsilon: se debe definir un valor entero positivo, que controle la diferencias entre un término y el anterior. Terminar cuando esa diferencia es despreciable (ínfima).

k: cantidad de veces a iterar. Básicamente se lo usa para control en el caso de que las diferencias no se cumplan (punto anterior).

33)Un oficial de correos decide optimizar el trabajo de su oficina cortando todas las palabras de más de cinco letras a sólo cinco letras (e indicando que una palabra fue cortada con el agregado de una arroba). Además elimina todos los espacios en blanco de más.

Por ejemplo, al texto " Llego mañana alrededor del mediodía " se transcribe como "Llego mañan@ alred@ del medio@".

Por otro lado cobra un valor para las palabras cortas y otro valor para las palabras largas (que deben ser cortadas).

1. Escribir una función que reciba un texto, la longitud máxima de las palabras, el costo de cada palabra corta, el costo de cada palabra larga, y devuelva como resultado el texto del telegrama y el costo del mismo.
2. Los puntos se reemplazan por la palabra especial "STOP", y el punto final (que puede faltar en el texto original) se indica como "STOPSTOP".

Al texto:

" Llego mañana alrededor del mediodía. Voy a almorzar "

Se lo transcribe como:

"Llego mañan@ alred@ del medio@ STOP Voy a almor@ STOPSTOP".

34)Distancia de Hamming.La efectividad de los códigos de bloque depende de la diferencia entre una palabra de código válida y otra. Cuanto mayor sea esta diferencia, menor es la posibilidad de que un código válido se transforme en otro código válido por una serie de errores.

A esta diferencia se le llama distancia de Hamming, y se define como el número de bits que tienen que cambiarse para transformar una palabra de código válida en otra palabra de código válida.

Si dos palabras de código difieren en una distancia **d**, se necesitan **d** errores para convertir una en la otra. Ejemplo: Entre los dos números binarios 0**1**0**1**0**1**0**1** y 0**0**0**0**1**1**1**1** hay una distancia de 4 bits, es decir, se necesitan cuatro errores para transformar un código en el otro.

a)Desarrollar un programa que dado dos números binarios de 8bits, indicar la distancia de Hamming correspondiente a los mismos.

b)Desarrollar un programa que dado 2 lista con números binarios de 8 bits, determine la distancia de hamming total existente entre ambas.

35)Matriz Caracol. Dada un número pasado por el usuario que será el tamaño, generar la matriz caracol a)Llenando de afuera hacia adentro de valores enteros en forma creciente. Ej:

```
   →              →
   1  2  3  4  5  6 ↓
  20 21 22 23 24  7
↑ 19 32 33 34 25  8
  18 31 36 35 26  9
  17 30 29 28 27 10
  16 15 14 13 12 11 ↓
   ←              ←
```

b) Llenando de adentro hacia afuera de valores enteros en forma creciente. Ej:

25 24 23 22 21
10 9 8 7 20
11 2 1 6 19
12 3 4 5 18
13 14 15 16 17

nota: la matriz debe ser cargada en un arreglo unidimensional.

36)Dada una matriz ingresada por el usuario con valores enteros, validar que sea cuadrada y que sea

a)Una matriz caracol llenada de afuera hacia adentro de valores enteros en forma creciente

b)Una matriz caracol llenada de adentro hacia afuera de valores enteros en forma creciente

Nota: la matriz a validar se encuentra en un arreglo unidimensional.