Implementa queda función solicitada en esta guia en un archivo .c aparte cuyo nombre debe ser idéntico al nombre de la función. Además cree un archivo .h donde deberá colocar todos los prototipos de las funciones implementadas. Finalmente realice un programa de testeo para cada una de las funciones implementadas demostrando su funcionamiento (use el .h creado aanteriormente). El nombre de cada programa de testeo debe ser guiaDeClase07_ejercicioNumero.c
Todos los archivos deberán ser subidos al repositorio dentro de una carpeta con el nombre guiaDeClase07 y deben estar comentados con doxygen

1. Implemente una función que me indique si el string ingresado contiene solo letras o solo números. El prototipo de la función es el siguiente:

```
int validaString (char *dataPtr);
```

   Devuelve
   - 1 si el string contiene solo letras .
   - 2 si el string contiene sólo números.
   - 0 Si no es ninguna de las anteriores.

2. Implemente una función que valide números de tarjeta de crédito utilizando el algoritmo de Luhn. El prototipo de la función es el siguiente:

```
int luhnAlg (char *tarjeta);
```

   Parámetros
   - tarjeta: puntero al vector que contiene el número de la tarjeta de crédito terminado en '\0'

   Devuelve:
   - Cero o un número positivo indicando que la tarjeta es válida.
   - -1: Cuando la cantidad de dígitos de la tarjeta es distinto que 16.
   - -2: Indica que el número de tarjeta es inválido.

   Algoritmo de Luhn
   a. Multiplique los dígitos que se encuentran en la posición impar del vector por dos, si el resultado es mayor o igual que diez se suman cada uno de los dígitos.
   b. Multiplique los dígitos que se encuentran en la posición par del vector por uno.
   c. Sume todos los resultados obtenidos en los puntos 1 y 2, obteniendo la suma llamada S. Si el módulo 10 de la suma obtenida S es igual a cero, el número de tarjeta es válido.

   Ejemplo:

Dígitos Tarjeta	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	
Valor a multiplicar Por dígito	X1	X2	X1	X2	X1	X2	X1	X2	X1	X2	X1	X2	X1	X2	X1	X2	
Resultado de la multiplicación	0	2	2	6	4	10	6	14	8	18	0	2	2	3	4	10	
Dígitos a sumar	0	2	2	6	4	1+0	6	1+4	8	1+8	0	2	2	6	4	1+0	= 58

   58%10 = 8 => La tarjeta es inválida

3. Implemente una función que realice la copia de un string hasta que llegue al '\0' o hasta que se copien los n caracteres indicados como parámetros. El prototipo de la función es

```
void myStrncpy (char *d, char *s, int n);
```

Donde:

s es el puntero al string origen
d es el puntero al vector destino
n es la cantidad de caracteres a copiar

4. Implemente una función que analice un párrafo de texto y devuelva la cantidad de caracteres y palabras que contiene. El prototipo de la función es el siguiente:

```
int analizaString (char *dataPtr, int *palabrasCant,
                   int *caracteresCant);
```

Donde:

dataPtr es el puntero al string a analizar
palabrasCant es un puntero a una variable donde se almacenará la cantidad de palabras del texto
caracteresCant es un puntero a una variable donde se almacenará la cantidad de palabras del texto

Devuelve

- 0 si el string contiene solo letras y signos de puntuación.
- -1 en caso contrario.

Notas de implementación:

- Los caracteres a contar son todas las letras del alfabeto, números, signos de puntuación y espacios.
- Una palabra se separa de otra por un espacio o signo de puntuación.
- No cuente espacios para obtener la cantidad de palabras, ya que el texto puede contener dos espacios seguidos y esto dará un conteo incorrecto.

5. Implemente una función que convierta un número positivo hexadecimal de 4 dígitos almacenado en un string y devuelva el correspondiente número decimal. Las letras del número hexadecimal estan en mayuscula.
El prototipo de la función es:

```
int hexaToDec (char *dataPtr);
```

Devuelve:

- El valor hexadecimal en decimal
- -1 Si hay un símbolo que no corresponda a un número hexadecimal.
- -2 Si la cantidad de dígitos es distinta a 4

Ejemplos

```
hexaToDec ("0001"); // Devuelve 1
hexaToDec ("CAFE"); // Devuelve 51966
hexaToDec ("JJJJ"); // Devuelve -1
hexaToDec ("1");    // Devuelve -2
```

6. Implemente una función que imprima en binario un número pasado como parámetro. El prototipo de la función es

```
void imprimirBinario (unsigned int n);
```

Donde:

n número a imprimir en binario.

7. Implemente una función que elimine de un string el carácter pasado como parámetro. Tenga en cuenta que al eliminar caracteres deberá ajustar la posición del '\0' El prototipo de la función es el siguiente.

```
int eliminarCaracter (char *dataPtr, char c);
```

Donde:

dataPtr: Es el puntero al string a modificar.

c: Carácter a eliminar.

Devuelve:

- La cantidad de caracteres eliminados.

8. Implemente una función que calcule el promedio de las notas pasadas por un string separadas por espacios. La última nota tiene un espacio también. El prototipo de la función es el siguiente

```
float calcularPromedio (char *dataPtr)
```

Si el string pasado como parámetro está mal formado devuelva NAN

Ejemplos:

```
calcularPromedio("Hola");                    // Devuelve NAN
calcularPromedio("");                        // Devuelve NAN
calcularPromedio("1 2 3 12");                // Devuelve NAN
calcularPromedio("0 1 2 3 4 5 6 7 8 9 ");    // Devuelve 4.5
```

Notas de implementación:

- Esta función solamente calcula el promedio de números de un digito separados por espacio.

9. Implemente una función que busque dentro de un string todas las ocurrencias de una palabras pasada como parámetro y las reemplace por otra palabra. El prototipo de la función es el siguiente:

```
int reemplazaPalabra (char *dataPtr, char *palabraBuscar,
                      char *palabraNueva);
```

Donde:

`dataPtr`: Es un puntero al string donde se realizan los reemplazos

`palabraBuscar`: Es un puntero a la palabra a buscar.

`palabraNueva`: Es un puntero a la palabra a reeemplazar

Devuelve

a. un número positivo indicando la cantidad de palabras reemplazadas..
b. -1 en caso de que `palabraBuscar y palabraNueva` tengan distinto tamaño.
c. -2 en caso de que `palabraBuscar y palabraNueva` sean `'\0'`.

10. Implemente una función que calcule la media móvil de un vector de n flotantes. El prototipo de la función es el siguiente:

```
void mediaMovil4(float *org, float *dest, int n);
```

Parámetros

- org: puntero al vector de flotantes al que hay que calcularle la media movil
- dest: puntero donde se almacena el resultado
- n: cantidad de elementos del vector origen

Algoritmo (tenga en cuenta que se explica con la sintaxis de vectores el algoritmo para que sea más simple, pero deberá usar punteros)

a. Para obtener el elemento dest[0] se calcula el promedio de los elementos 0,1, 2, 3 del vector org.
b. Para obtener el elemento dest[1] se calcula el promedio de los elementos 1,2, 3, 4 del vector org.
c. Se continúa así hasta llegar al último elemento del vector org.

Ejemplo:

índice	0	1	2	3	...	n-4	n-3	n-2	n-1
dest	(org[0] + org[1] + org[2] + org[3]) /4	(org[1] + org[2] + org[3] + org[4]) /4	(org[2] + org[3] + org[4] + org[5]) /4	(org[3] + org[4] + org[5] + org[6]) /4		(org[n-4] + org[n-3] + org[n-1] + org[n-1]) /4	0	0	0

11. Implemente una función cuente la ocurrencia de cada carácter de un string pasado como parámetro.

```
void contarCaracteres (char *dataPtr, int *dataCntPtr);
```

Donde:

`dataPtr`: Es el puntero al string en el que hay que contar la ocurrencia de cada carácter.

`dataCntPtr`: Es el puntero un vector de 256 enteros en el que se lleva la cuenta de los caracteres del string

Ejemplo:

El string apuntado por dataPtr es “11AB1B1ZZZZ1”

- La posición 65 (65 es el ASCII de la ‘A’) del vector apuntado por `dataCntPtr` debe tener el número 1 (Cantidad de ‘A’ en el string)
- La posición 66 (66 es el ASCII de la ‘B’) del vector apuntado por `dataCntPtr` debe tener el número 2 (Cantidad de ‘B’ en el string)
- La posición 90 (90 es el ASCII de la ‘Z’) del vector apuntado por `dataCntPtr` debe tener el número 4 (Cantidad de ‘Z’ en el string)
- La posición 49 (49 es el ASCII de la ‘1’) del vector apuntado por `dataCntPtr` debe tener el número 5 (Cantidad de ‘1’ en el string)
- El resto de los elementos del vector deberán estar en cero.

12. Implemente una función que obtenga el dígito verificador de un número de CUIT pasado como parámetro, el cálculo se realiza utilizando el algoritmo módulo11. El prototipo de la función es el siguiente:

```
int cuitValida (char *cuit);
```

Parámetros

- cuit: puntero al vector que contiene el número de CUIT terminado en '\0'

Devuelve:

- Un número positivo indicando el dígito verificador.
- -1: Cuando la cantidad de dígitos es distinto de 10
- -2: Indica que el número de CUIT es inválido (contiene algo distinto a números)

Algoritmo módulo 11

a. Multiplique los dígitos Desde el menos significativo por la serie 2,3,4,5,6,7.
b. Sume el resultado de las multiplicaciones anteriores.
c. Calcule el módulo 11 de la suma anterior.
d. Al resultado anterior restele 11, si el resultado es menor que 10 lo obtenido es el dígito verificador. En cambio si vale 10 el dígito verificador es 9. Si vale 11 el dígito verificador es 0

Ejemplo:

CUIT	2	0	1	2	3	4	5	6	7	8	Suma	%11	Dígito
Valor a multiplicar Por dígito	X5	X4	X3	X2	X7	X6	X5	X4	X3	X2		148%11	11-5
Resultado de la multiplicación	10	0	3	4	21	24	25	24	21	16	=148	=5	6