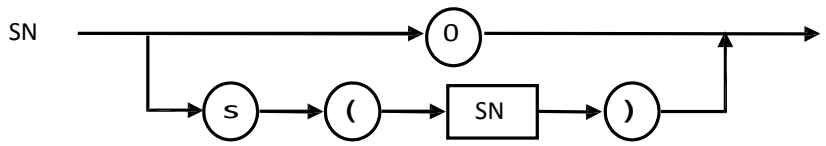
Ejercicios de Recursividad -

Ejercicio 1 . Los números naturales pueden ser representados como una función unaria usualmente llamada s^n.
Por ejemplo:



0 = 0
1 = s(0)
2 = s(s(0))
3 = s(s(s(0)))
7 = s(s(s(s(s(s(s(0)))))))

a. Realice un procedimiento recursivo que reciba un entero positivo y muestre por pantalla la representación de dicho número en formato $s^n(0)$.
b. Realice una función recursiva que reciba una secuencia de caracteres ingresada por teclado representando un número en notación $s^n(0)$ retorne el entero correspondiente.

Ejercicio 2. Escriba el planteo recursivo e implemente en C los siguientes ejercicios teniendo en cuenta las restricciones impuestas para cada caso. Obviamente, podrán ser utilizadas en todos los casos las estructuras de control y llamadas a funciones necesarios siempre y cuando los mismos también respeten las restricciones antes mencionadas.

a) Una función recursiva suma: N x N →N utilizando solamente como primitivas `succ` y `pred`.
b) Una función recursiva resto: N x N →N que obtenga el resto (módulo) de la división entera utilizando como única operación aritmética la resta (no puede usarse div). Ej.: resto(5,2) = 1, resto(8,2) = 0, resto(1,2) = 1.
c) Una función recursiva divEntera: N x N →N que obtenga el cociente (resultado) de la división entera utilizando como únicas operaciones aritméticas la suma y la resta.
d) Una función recursiva cuadrado: N →N que obtenga el cuadrado de un número natural distinto de cero utilizando exclusivamente el siguiente método: el cuadrado(k) es igual a la suma de los k primeros números impares. Por ejemplo, el cuadrado de 4 es 1+3+5+7=16.

Ejercicio 3. Escriba un planteo recursivo e implemente ambos ejercicios en C:

a) Mostrar los números del 1 al N en orden creciente.
b) Mostrar los números del 1 al N en orden decreciente.

Ejercicio 4 . Escriba un planteo recursivo e implemente en C los siguientes ejercicios:

a) Una función recursiva que determine si un dígito D no pertenece a un número entero positivo N. Ej.: si N=1323 y D=5 el resultado es Verdadero, y si D=1 el resultado es Falso.
b) Una función que cuente la cantidad de dígitos pares en un número entero. Ej.: si el número es 22005 el resultado es 4, y si fuera 35 el resultado es 0.
c) Una función que cuente la cantidad de dígitos pares que ocupan posiciones impares (identificándolas de izquierda a derecha) en un número entero. Ej.: si el número es 22005 el resultado es 2, y si fuera 1414 el resultado es 0

Ejercicio 5 . Considere la siguiente función recursiva:

```
int misterio(int a, int b){
    if (b == 0) then misterio= 0;
    else if (b%2 == 0) then misterio= misterio(a+a, b/2);
         else misterio= misterio(a+a, b/2) + a;}
```

a) Realice una traza para las siguientes llamadas:
   i. `misterio(2,25);`
   ii. `misterio(3,11);`
b) Determine que función matemática define `misterio`.
c) Si se reemplaza en la línea 3 por `misterio= 1` y las operaciones + por * en las líneas 4 y 5, ¿Qué función matemática queda definida?

Ejercicio 6. Escriba un planteo recursivo e implemente en C los siguientes ejercicios:

a) Una función recursiva que determine si un número natural es potencia de 2. Ej.: espot2(33)=false, espot2(64)=true.
b) Una función recursiva que determine si dígito D está ubicado en la posición más significativa de un número natural. Ej.: pmasS (2,2345) = true, pmasS (6,5604) = false, pmasS (7,945) = false.

c) Una función recursiva que determine si un número natural P es prefijo de un número natural Q. Ej.: esPrefijo(25,2545)= true, esPrefijo(4,5604)= false, esPrefijo(459,45)= false, esPrefijo(25,25)= true.
d) Una función recursiva que determine si un número natural P es sufijo de un número natural Q. Ej.: esSufijo(25,2545)= false, esSufijo(4,5604)= true, esSufijo(459,45)= false, esSufijo(25,25)= true.

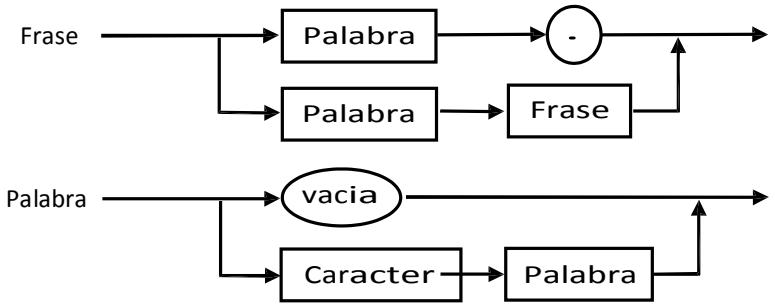
Ejercicio 7. Realice una traza suponiendo que se produce la siguiente llamada al procedimiento recursivo ex237(6) y muestre la información que se imprimirá en pantalla como resultado de su ejecución.

```
void ex237(int n){
if (n>0) { printf("%d",
n);
ex237(n-2);
ex237(n-3);
printf("%d", n); }}
```

Ejercicio 8. Escriba un planteo recursivo y defina funciones recursivos para cada caso:

a) Leer una cadena de caracteres de longitud arbitraria finalizada en # y mostrar la cadena en orden inverso. Ej.: si se tipea animal# deberá imprimirse en pantalla lamina
b) Leer una cadena de caracteres de longitud arbitraria finalizada en # y mostrar la cadena en orden inverso sin mostrar las vocales. Ej.: si se tipea animal# deberá imprimirse en pantalla lmn

Ejercicio 8. Defina funciones recursivas que permitan leer una frase terminada en punto y contar la cantidad de palabras de dicha frase. Por ejemplo, si la frase ingresada es "Que lindo día." deberá devolver 3, y si la frase fuera "." deberá devolver 0.



Ejercicio 9. Escriba el planteo recursivo e implemente en C una función recursiva que calcule la suma de los dígitos que ocupan posiciones impares para un número natural. Se considera que la posición 1 es la posición del dígito menos significativo (lugar de la unidad), la posición 2 es la posición de la decena, etc. Por ejemplo, si se considera el natural 587, el 7 está en la posición 1, el 8 en la posición 2 y el 5 en la posición 3. En el ejemplo, la función debería retornar 12 (7+5).

Ejercicio 10. Dado un número natural, definiremos como su número promedio al número que se obtiene de sumar sus dígitos impares y restar sus dígitos pares. Por ej.: el número promedio de 318547 es 4 esto es, numeroPromedio(318547) = numeroPromedio(31854)+7 = numeroPromedio(3185) - 4 + 7 = …
Escriba el planteo recursivo e implemente en C una función que obtenga su número promedio.

Ejercicio 11. Escriba un planteo y una función recursiva para imprimir una media pirámide de dígitos como se muestra en la siguiente figura. Utilice un procedimiento recursivo para generar cada fila de la media pirámide.

```
1
21
321
4321
54321
654321
7654321
87654321
987654321
```

Ejercicio 12. Dada una secuencia de números enteros positivos finalizada en -1 (el cual no se considera parte de la misma), escribir un planteo recursivo y la correspondiente implementación para:

a) Sumar todos los enteros de dichas secuencia. Ej.: Para la secuencia 2 5 3 6 12 3 -1 el resultado es 31.
b) Mostrar por pantalla todos los valores de la secuencia que sean divisibles por el último valor de la misma. c) Calcular el promedio de los valores de la secuencia.
d) Determinar el k-ésimo elemento de la secuencia comenzando desde adelante. El valor k debe ser proporcionado por el usuario. Ej.: Para la secuencia 2 5 3 6 12 3 -1 y k = 4 el resultado es 6.