TPN°9: Recursividad

En esta práctica ninguna función debe tener más de 10 líneas de código

Ejercicio 1

Escribir una función recursiva que reciba como parámetros de entrada un vector de enteros y su dimensión, y que devuelva en su nombre la suma de todos sus elementos.

Ejercicio 2

Escribir la función recursiva **suma** que reciba un único parámetro de entrada de tipo integer y devuelva en su nombre la suma de sus dígitos.

Por ejemplo, **suma**(3498) devuelve 24

Ejercicio 3

Escribir una función recursiva que reciba tres parámetros de entrada representando dos vectores de números reales de igual tamaño y su dimensión. La función debe regresar en su nombre el producto escalar de los vectores recibidos.

Ejercicio 4

Se desea escribir funciones que dado un número entero mayor o igual a cero, indiquen si el mismo es par o impar. ¿Qué ocurre si se las define de la siguiente forma?

```
int esPar(unsigned n);
int esImpar(unsigned n);

int
esPar(unsigned n)
{
      return ! esImpar(n);
}

int
esImpar(unsigned n)
{
      return ! esPar(n);
}
```

Ejercicio 5

Idem ejercicio anterior pero con las siguientes funciones. ¿En qué casos funcionan correctamente y en qué casos no?

```
int esPar(unsigned n);
int esImpar(unsigned n);

int
esPar(unsigned n)
{
    if (n = = 0)
        return 1;
    else
        return esImpar(n-1);
}

int
esImpar(unsigned n)
{
    if (n = = 1)
        return 1;
    else
        return esPar(n-1);
}
```

Ejercicio 6

Escribir una función que reciba como parámetro de entrada un string y retorne 1 si el mismo es palíndromo y 0 en caso contrario. Resolver el problema en forma recursiva

Ejercicio 7

Escribir una función recursiva ***busquedaBinaria*** que reciba como únicos parámetros:

- **datos**: un vector de enteros ordenados en forma ascendente.
- **dim**: la dimensión del mismo
- **num**: un número entero

Dicha función debe devolver 1 si **num** es un elemento de **datos** y 0 en caso contrario. La búsqueda del elemento se debe realizar utilizando el algoritmo de ***búsqueda binaria.***

Ejercicio 8

Calcular por medio de una función recursiva que reciba como parámetros de entrada dos números enteros, la fórmula de ACKERMANN correspondiente a los mismos, de acuerdo a las siguientes reglas:

- ACK (0,N) = N + 1
- ACK (M,0) = ACK (M - 1, 1)
- ACK (M,N) = ACK (M - 1, ACK (M , N - 1))

Ejercicio 9

El método de Newton-Raphson para aproximar la raíz cuadrada de un número positivo se puede calcular aplicando sucesivamente los siguientes pasos un cierto número de veces ITER:

$$\mathbf{valorAproximado} = \frac{\mathbf{valorAnterior} + \frac{\mathbf{X}}{\mathbf{valorAnterior}}}{\mathbf{2}}$$

donde el valor original conveniente para comenzar este ciclo suele ser X/2.

Escribir una función recursiva ***raizCuadrada*** que regrese en su nombre el valor aproximado de la raíz cuadrada de un número, recibiendo tres parámetros de entrada:

- **valorAnterior**: real (representa el valor hasta ahora candidato como raíz cuadrada)
- **iter**: entero que representa la cantidad de iteraciones que deben realizarse para aproximar la raíz cuadrada.
- **x**: real al que se le quiere calcular la raíz cuadrada.

Ejemplo:

Si valorAnterior = 4, iter= 3, x= 8, entonces la función devolverá 2.8333 (los valores sucesivos obtenidos son: 4, 3, 2.833)

Ejercicio 10

Escribir una función recursiva que reciba dos parámetros de tipo string, uno de entrada y el otro de salida. La función debe devolver en el segundo string los caracteres del primero en forma invertida. En la primera invocación de esta función recursiva el segundo string debe contener todos los caracteres en cero y con el suficiente espacio como para almacenar al primero, pero no debe validar estas condiciones.

Ejemplo:

```
char origen[] = "Practica";
char destino[20] = {0};

invierte(origen, destino);
```

Con esta invocación, en **destino** se obtendrá el string **"acitcarP".**

Ejercicio 11

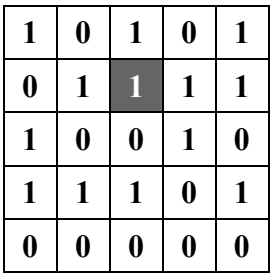
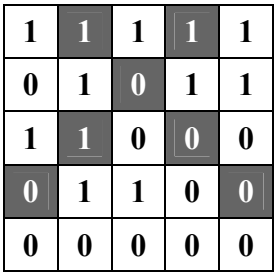
Escribir una función que reciba cuatro parámetros:

- Matrix: parámetro de entrada-salida que representa una matriz cuadrada booleana
- Dim: parámetro de entrada de tipo entero que representa la dimensión de la matriz
- Fila: parámetro de entrada de tipo entero que representa un número de fila de la matriz
- Columna: parámetro de entrada de tipo entero que representa un número de columna de la matriz

La función se encargará de devolver dicha matriz cambiada de la siguiente forma: el elemento de la fila y columna indicada y todos los de las dos diagonales que pasen por él se reemplazarán por su negación booleana, los demás elementos de la matriz quedan igual .

Esta función deberá ser recursiva o deberá invocar a otra función auxiliar recursiva para realizar lo pedido.

Ejemplo:

Si se recibe la siguiente matriz de 5x5 junto a los valores 1 (fila) y 2 (columna)	Se debe devolver la matriz cambiada así:
	

Ejercicio 12

Programar una ***función recursiva*** BALANCE que reciba como único parámetro un string constante representando una expresión matemática. Dicha función debe devolver el valor 0 si hay igual cantidad de paréntesis que abren y que cierran, y retorna un valor distinto de cero en caso contrario. **No definir variables ni funciones auxiliares**.

Ejemplos

si la función Balance recibe	*entonces devuelve*
`"56 + ( a – 7 – ( 12 + c ) – t * 678 )"`	**0**
`"56 + ( a – 7 – 12 + c ) – t * 678 )"`	**distinto de 0**
`" + ( a ( 10 + c * 789 ) – 6 / 8 ) + 40"`	**0**
`"5 + 2 – 7 )"`	**distinto de 0**
`")5 + (2 – 7"`	**0**