T.P. 00: (básico)

Hacer un programa que realice las siguientes operaciones básicas de:

- Suma
- Resta
- Multiplicación
- División
- Potencia

Utilizando una función para caa una de las operaciones y validando los valores en los casos que sean necesarios.

T.P. 01: (Recursividad)

Hacer un programa que permita la carga de un vector de 15 elementos enteros, con valores seleccionados al azar entre un rango de números ingresados por el usuario, mostrar el vector y hacer una función que de forma recursiva ordene el vector de mayor a menor.

T.P. 02: (Punteros – Matrices - Colores)

Hacer un programa que permita la creación de dos matrices a partir del ingreso de los datos sobre la cantidad de filas y columnas para ambas, y muestre las matrices en forma de matriz, y el resultado de la multiplicación elemento a elemento, debiendo destacar los elementos de ambas matrices que se van multiplicando.

T.P. 03: (Tipos y Switch)

Definir un registro con los siguientes campos enteros:

- X
- Y
- Z
- Dim

Donde (x, y, z) representan las posibles coordenadas de un punto y (Dim) indica cuantas de esas coordenadas serán utilizadas. El programa deberá pedir:

1) La cantidad de dimensiones con que se desea trabajar.
2) Los pares de coordenadas de 2 (dos) puntos, y devolverá la distancia entre ambos puntos.

T.P. 04: (Recursividad)

Hacer un programa que permita indicar si un número es divisible por tres, utilizando una función recursiva que recibe como parámetro un valor entero y devuelve 0 (cero) si el divisible o 1 (uno) si no lo es.

```
int esDivisiblePorTres ( int )
```

Pista: Cuando un número es divisible por tres, la suma de sus dígitos también lo es.

T.P. 05: (Recursividad)

Hacer una función recursiva que dado un entero sin signo, devuelva, según el parámetro entero, el valor del término de la **SUCESIÓN** o el resultado de la **SERIE** de Fibonacci. La función deberá tener el siguiente prototipo:

unsigned long int fibonacci (unsigned int, int);

Sabiendo que para calcular el término n de la sucesión se utiliza:

$$F(n) = \begin{cases} 0 & \text{Si } n = 1 \\ 1 & \text{Si } n = 2 \\ F(n-1) + F(n-2) & \text{Si } n > 2 \end{cases}$$

Y que la serie hasta el termino n es:

$$F(n) = \sum_{i=0}^{n} F(i)$$

donde **F(n)** es igual a la suma de todas los llamados a **F(i)** con **i** desde **0** hasta **n**.

El programa deberá mostrar para el primer caso solo el resultado de la sucesión en el término solicitado, mientras que para el segundo caso deberá mostrar todos los valores involucrados en la operación.

T.P. 06 (+ Recursividad - BackTracking)

Hacer un programa que permita generar una matriz en N x M entera, y que al azar, la cargue con valores 0 (ceros) y 1 (unos). Esos valores van a representar paredes, para los unos y caminos para los ceros. El programa deberá encontrar todas las salidas del laberinto, teniendo como condición que siempre la entrada al laberinto se encuentra en la primera fila y la salida del mismo en la última.

0	**1**	**0**
1	**0**	**1**
0	**1**	**0**

En éste ejemplo existen dos entradas al laberinto y dos salidas, y en total hay ocho caminos posibles.

(0,0)(1,1)(2,2) >> (0,0)(1,1)(1,0) >> (0,0)(1,1)(0,1)(1,1)(1,0) >> (0,0)(1,1)(0,1)(1,1)(0,1) …

El desplazamiento es en cualquiera de los ocho direcciones.

T.P. 07 (+ Recursividad - BackTracking)

Hacer un programa que permita el ingreso de una palabra de 16 caracteres, la distribuya en una matriz de la misma cantidad de filas y columnas de manera que las filas pares se lean de izquierda a derecha, mientras que las impares de derecha a izquierda. Además almacenar cinco palabras de hasta dieciséis caracteres, que mediante un algoritmo recursivo deberá buscar si es posible formar cada una de las palabras con las letra de la matriz, teniendo en cuenta que las letras se encuentren adyacentes y nunca una letra debe utilizarse más de una vez para la formación de la palabra, debiendo mostrar la cantidad de veces que aparece la palabra y las coordenadas que la forman.

Ejemplo: **C O N S T I T U C I O N A L E S**

C	**O**	**N**	**S**
U	**T**	**I**	**T**
C	**I**	**O**	**N**
S	**E**	**L**	**A**

CONSTITUCIONALES	**1**
TITO	**3**
LEONA	**1**
SITIO	**3**
HELP	**0**

- Donde **CONSTITUCIONALES** se forma una vez.
- **TITO** 3 veces (1,1)(1,2)(1,3)(2,2) // (1,3)(1,2)(1,1)(0,1) // (1,3)(1,2)(1,1)(2,2)
- **LEONA** 1 vez (3,2)(3,1)(2,2)(2,3)(3,3)
- **SITIO** 3 veces (0,3)(1,2)(1,1)(2,1)(2,2) // (3,0)(2,1)(1,1)(1,2)(2,2) // (3,0)(2,1)(1,1)(1,2)(0,1)

T.P. 08 (Recursividad)

Hacer un programa que permita el ingreso de un valor entero sin signo que indica el número del nivel, y utilizando una función recursiva muestre el siguiente cálculo:

Ejemplo: Nivel = 5

0	1
1	1 1
2	1 2 1
3	1 3 3 1
4	1 4 6 4 1
5	1 5 10 10 5 1

Debe tener en cuenta que siempre el primer nivel es un 1(uno), cada nivel se forma sumando los números del nivel anterior y que en los extremos siempre hay 1 (unos).

Nota:

- Los trabajos prácticos deberán ser guardados en una carpeta "**Trim1**" y dentro de ella cada programa llevará el nombre "**labo22gxx-tpnn.c**" (ejemplo: labo22a01-tp01.c).