Práctica 12.- Funciones Profundidad y eliminar sub-expresión.

- **Objetivo de la práctica:** Desarrollar alguna función más relacionada con la práctica obligatoria III. Las funciones aquí pedidas, pueden ser utilizadas como funciones auxiliares de otras de mayor complejidad, permitiendo que éstas reduzcan su complejidad.

Ejercicio 1.- Calcular la profundidad de una expresión.

En este primer ejercicio vamos a programar una función que denominaremos *Depth*, que nos permita calcular la profundidad que tiene una expresión que reciba como parámetro.

Esta función recibirá un único parámetro tipo *Expr*, que será la expresión a analizar, y devolverá como resultado una nueva expresión con valor Head='Symbol' y Terminal='n', siendo n la profundidad calculada de la expresión que ha recibido como parámetro. Veamos algún **ejemplo:**

Depth[{}] ➔ Profundidad 1
Depth[{a,b,c,d}] ➔ Profunidad 2
Depth[{a,{b,c},{{{e},f},r}}] ➔ Profundidad 5

La función debe ser declarada y programada en la *Unit Ejercicios*, de la siguiente forma:

function Depth(A : Expr) : Expr;

Ejercicio 2.- Eliminar sub-expresión de una expresión dada.

En este segundo ejercicio, debéis programar una función que denominaremos *RemoveAll*, que nos permita eliminar en el primer nivel bajo la raíz de la primera expresión pasada, todas las apariciones de la sub-expresión.

Por lo tanto, esta función recibirá 2 parámetros X e Y de tipo *Expr*: X será la expresión a la que queremos eliminar sub-expresiones, e Y la sub-expresión que debemos buscar en los hijos de la raíz de X para eliminar.

La búsqueda y posterior eliminación de las apariciones de Y en X, sólo debe efectuarse en el primer nivel bajo la raíz. Veamos algún **ejemplo**:

RemoveAll[{a,{a,b},{{a},c}} ➔ Resultado: {{a,b},{{a},c}}
RemoveAll[Sen[x,y,z],y] ➔ Resultado: Sen[x,z]
RemoveAll[{nos,hola,despedimos,hola,del,hola,curso},hola] ➔{nos,despedimos,del,curso}

Si la sub-expresión buscada Y aparece en otro nivel diferente en X al de los hijos de la raíz, no debemos eliminarla. La declaración de la función en la *Unit Ejercicios* será como sigue:

function RemoveAll(X : Expr; Y : Expr) : Expr;

Propuesta de mejora de la función: Como mejora a esta función se puede programar, sin mucho esfuerzo, una segunda función que puede denominarse RemoveAll2, que elimine todas y cada una de las apariciones de la sub-expresión Y en la expresión X. Para poder ejecutar esta nueva función, además de ser programada en la *Unit Ejercicios*, es necesario retocar la *Unit Kernel*, añadiendo entre sus líneas de código un nuevo "else" con la aparición en el intérprete de comandos de esta función.

Declaración de la función en la *Unit Ejercicios*:

function RemoveAll2(X : Expr; Y : Expr) : Expr;

Líneas de código necesarias en la *Unit Kernel*. Deben ser añadidas después del "else" que permite llamar a la función *RemoveAll*:

```
else if (Ex^.Head = 'RemoveAll2') then
        EvaluateExpr:=RemoveAll2(SubExpr(Ex,1), SubExpr(Ex,2))
```