Defensa Práctica III - Junio 2010

Ejercicio 1.- Insertar primer nivel bajo la raíz a una expresión.

- El ejercicio consiste en programar (al estilo de las prácticas semanales de laboratorio) una función con la siguiente cabecera:

function AddFirstLevel(X : Expr; var ec : TException) : Expr;

- En vuestros respectivos escritorios, disponéis de una carpeta denominada DefensaJunio2010. La función se encuentra declarada en la Unit Ejercicios.pas. En dicho fichero debéis resolver y guardar la solución al ejercicio.

- El cometido de la función debe ser el de crear una nueva expresión a partir de la expresión X pasada, insertando un nuevo primer nivel bajo la raíz que cumpla las siguientes directrices: Los hijos "impares" de la expresión X original, pasarán a tener un nuevo padre que será la función "Sen". Para los hijos "pares" de X, su nuevo "padre" será la función "Cos". Todas las funciones "Sen" y "Cos" formarán el nuevo primer nivel de la expresión devuelta. **Ejemplo: X={a,b,c,d,e} AddFirstLevel[X] = {Sen[a],Cos[b],Sen[c],Cos[d],Sen[e]}.**

- Las expresiones pasadas a la función, deberán cumplir la siguiente regla, provocando en caso contrario el error a continuación descrito:

    1. **La expresión pasada debe ser siempre una lista o una función**. En caso contrario, la función deberá devolver el número de error 1, acompañado del mensaje de error "La función sólo trabaja sobre listas y funciones.". Ejemplo:

        - ✓ AddFirstLevel[León] ➔ Error 1.

- En la carpeta que contiene el código fuente de la práctica, tenéis disponible una batería de prueba con los resultados esperados. A continuación se muestran algunos ejemplos de lo que debe devolver la función a desarrollar:

    - ✓ AddFirstLevel[Sen[x,y,z]] ➔ Sen[Sen[x],Cos[y],Sen[z]]
    - ✓ AddFirstLevel[{{a,b},x,Cos[p]}] ➔ {Sen[{a,b}],Cos[x],Sen[Cos[p]]}
    - ✓ AddFirstLevel[{{x,y,{g,y},u},Cos[a,b,Sen[r,s]]}] ➔ {Sen[{x,y,{g,y},u}],Cos[Cos[a,b,Sen[r,s]]]}

Notas:

- **No** pueden aparecer como parte de la solución final las funciones **ExprToStr y ParseExpr**.
- **Obligatorio** un correcto funcionamiento con las opciones de compilación –Cr y –gh.
- Tiempo para realizar el ejercicio: **75 minutos.**

```
function AddFirstLevel(X : Expr; var ec : TException) : Expr;

var
        Ki:TExprIt;
        z,t:Expr;
        iteracion:integer;

begin

if (X<>nil) then begin
        if (X^.head='Symbol') then begin  //Si es un símbolo, devolvemos el error.
                        ec.nError:=1;
                        ec.msg:='La función no trabaja sobre símbolos';
                        AddFirstLevel:=nil; end
        else begin
        //No es símbolo. Añadimos la raíz a la solución, y vamos recorriendo la expresión X añadiendo
       // cada hijo a la solución con un nodo "Sen" o "Cos" según sea su posición par o impar en la
      // expresión X.
                AddFirstLevel:=AllocExpr(X^.head,X^.terminal);
                MoveToFirstSubExpr(X,Ki);
                iteracion:=1;
                while (IsAtNode(Ki)) do begin
                        t:=ExprAt(Ki);
                        if (iteracion mod 2 = 0) then begin
                                z:=AllocExpr('Cos','');
                                AddSubExpr(DeepCopy(t),z);
                                AddSubExpr(z,AddFirstLevel);
                        end
                        else begin
                                z:=AllocExpr('Sen','');
                                AddSubExpr(DeepCopy(t),z);
                                AddSubExpr(z,AddFirstLevel);
                        end;
                        iteracion:=iteracion+1;
                        MoveToNext(Ki);

                end;
        end;

  end;

end;
```