1. Definir la estructura de datos necesaria para implementar una Tabla Hash con los siguientes requisitos:
   a. Manejo de colisiones externo mediante listas dinámicas.
   b. La Tabla deberá almacenar como clave Tipo + Nro de Factura. El Tipo se compone de una letra y el número desde 1 a 50000.
   c. Encontrar la función Hash correspondiente para almacenar las facturas.
   d. El total de facturas a almacenas no supera las 30000.

Estructura de datos

```
Const
      MIN = 0;
      MAX = 30000;
      PosNula = 0;
      NroPrimo = 29999;

Type

      TipoElemento = record
         TipoFactura : Char;
         NumeroFactura : LongInt;
         NroRegistro : LongInt;
      End;
      Posicion_Tabla = LongInt;
      TipoRegistroTabla = Record
         Clave : TipoElemento;
         Ocupado: Boolean;
         ListaColision: Lista;
      End;
      TablaHash = Record
         Tabla: Array [MIN..MAX] of TipoRegistroTabla;
         Ocupados: Integer;
         CantidadClaves: Integer;
End
```

Función hash

```
Function FuncionTransformacion(X: TipoElemento): Posicion_Tabla;
Var i:LongInt;
Begin
      i:= ord(X.TipoFactura) * 100000;
      i:= i + X.NumeroFactura;
      FuncionTransformacion := (i Mod NroPrimo);
End;
```

2. Definir claramente que es: rama, altura, nivel y grado de un árbol.

***Rama* es un camino desde el nodo raíz a una hoja**
***Altura* es el máximo número de nodos de las ramas del árbol.**
***Nivel* de un nodo, es el número de nodos del camino desde la raíz hasta dicho nodo.**
***Grado* es el número máximo de hijos que tienen los nodos del árbol.**

3. Realizar un algorítmo que permita obtener el grado de un árbol n-ario. La estructura de datos es la vista en clase y se lo considera ya transformado en binario. La función deberá ser ***genérica recursiva*** y se invocará como *"GradoArbolNario (A : Arbol) : Integer"*

```
Function GradoArbolNArio (A:Arbol):Integer;
Var g: Integer;
   Procedure Hermanos (P:PosicionArbol, CH: Integer);
   Begin
         If not RamaNula(P) Then
         Begin
               Hermanos(HijoIzquierdo(A, P),0);
               If RamaNula(HermanoDerecho(A,P) Then
               Begin
                    If (CH + 1) > g Then
                       g:= CH + 1;
               end
               else
               Hermanos (HermanoDerecho(A,P), CH +1);
         End;
   End;
Begin
   g:=0:
   Hermanos (A.Raiz, 0);
   GradoArbolNArio:= g;
End;
```

4. Encontrar la cantidad mínima de claves para un Arbol AVL (binario balanceado) de tal forma de realizar en el siguiente órden las rotaciones RII (Izq. Simple), RDI (doble der-izq.), RDD (der. Simple) y RID (soble izq-der.). Se deberán mostrar la lista de claves para cumplir con lo arriba expuesto y además mostrar el dibujo correspondiente de cada rotación.

La cantidad mínima de claves a insertar para lograr las rotaciones requeridas en el orden requerido es 7. Por ejemplo: 20, 10, 5 (aquí se produce RII), 30, 25 (en este punto se debe efectuar RDI), 40 (para obtener RDD) y 35 (para concluir con RID)