P1. (2 ptos) Dada la siguiente gramática para generar arboles binarios. Se pide: a) ¿ es LL(1)?; b) construye la tabla de análisis sintáctico para un analizador descendente dirigido por tabla; c) realiza la traza para la cadena de entrada (4) (2 null null) (5 null null)). En caso de encontrar un error, emite el mensaje correspondiente, indicando el token que se ha encontrado y la lista de tokens que se esperaba. Aplica el método de recuperación en modo de pánico.

BTree → **(num** BTree BTree **)** | **null**

P2. (2 ptos) Dada la construcción condicional de los lenguajes de programación donde la parte *else* puede ser opcional

S → **if** C **then** S **else** S
S → **if** C **then** S
S → **other**
C → **and** C C | **0** | **1**

Indica si se trata de una gramática SLR(1). En caso de que existan conflictos, indica las acciones a realizar para que el operador *and* sea asociativo por la izquierda y la parte *else* esté asociada a su *if* más cercano. Notad que el operador *and* se encuentra en notación prefija en la condición.

P3. (1 ptos) Dada la siguiente gramática indica si es LL(1), SLR(1), y/o LR(1). Justifica tu respuesta.

E → A | B
A → **a** | **c**
B → **b** | **c**

P4 . (2 ptos) Diseña un ETDS que traduzca consultas sencillas en SQL escritas en inglés a castellano. La gramática que genera este tipo de consultas en SQL es:

Consult → ClausulaSelect ClausulaFrom ClausulaWhere
ClausulaSelect → **select id** (**,** **id**)*
ClausulaFrom → **from id** (**,** **id**)*
ClausulaWhere → **where** E
E → E **==** E | E **>** E | **id** | **num**

Por ejemplo: la entrada `select Nombre, NIF from Empleados where Edad>30` se convierte en `seleccionar Nombre, NIF desde Empleados donde Edad>30` . ¿ Qué tipo de atributo(s) es? Indica la estructura del árbol que representa este tipo de código. Implementa en pseudocódigo la función para su evaluación.

P5. (3 ptos) Supongamos la construcción de los lenguajes de programación que genera asignaciones según se cumpla una condición u otra:

S → **id =** E_1 **?** E_2 **:** E_3 // si E_1 es verdadero se asigna el valor E_2 sino E_3
E → **id**

a) indica el diagrama de flujo de esta construcción; b) indica la forma del árbol abstracto de análisis sintáctico que usarías para su traducción a código de 3 direcciones; c) implementa el pseudocódigo de la función generar_código para traducir estas sentencias; d) construye el árbol y traduce el siguiente código. Usa los operadores habituales vistos en clase.

```
a  =  b < c ? 2 : 3
d =  b > c ? 1 : 0
```