Departamento de Tecnologías de la Información

Área de Ciencias de la Computación e Inteligencia Artificial

Procesadores de lenguajes

Examen de septiembre

EJERCICIO 1 (2 puntos)

La siguiente figura muestra una expresión regular formada por los símbolos **a**, **b**, **c** y **d**.

(a|b)* ((c|d) (a|b)* (c|d) (a|b)*)* (c|d) (a|b)*

Obtenga el Autómata Finito Determinista asociado, indicando el conjunto de expresiones regulares punteadas que describen cada estado del autómata.

EJERCICIO 2 (2.5 puntos)

La siguiente gramática permite describir la estructura de una clase:

Clase → **class id llaveab** *ListaComp* **llavece**

ListaComp → *ListaComp Comp*

ListaComp → *Comp*

Comp → *Tipo* **id pyc**

Comp → **id parab** *ListaTipo* **parce pyc**

Comp → *Tipo* **id parab** *ListaTipo* **parce pyc**

ListaTipo → *ListaTipo* **coma** *Tipo*

ListaTipo → *Tipo*

ListaTipo → λ

Tipo → **int**

Tipo → **char**

(a) Construya la tabla SLR de la gramática planteada.

(b) Desarrolle la traza del analizador SLR para la siguiente cadena:

class id llaveab char id parab int coma int parce llavece

EJERCICIO 3 (3 puntos)

La siguiente gramática permite describir los pedidos de una cafetería.

Pedido → *Lista*

Lista → *Término* (**plus** *Término*)*

Término → **producto**

Término → **num** **prod** *Término*

Término → **lpar** *Lista* **rpar**

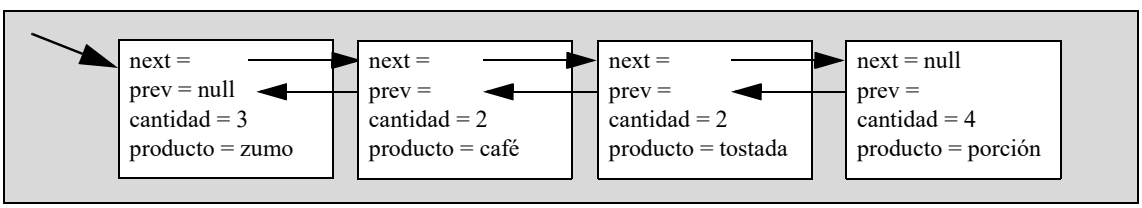
Para representar estos pedidos se pretenden utilizar listas de términos doblemente enlazadas. Cada uno de estos términos representa un producto del pedido y la información incluye el número de unidades de ese producto (por ejemplo, tres cafés) y se describe por medio de un objeto de la clase *Term*.

```
public class Term {

   private double cantidad;
   private String producto;
   private Term next;
   private Term prev;

   public Term(double c, String p) {
     this.cantidad = c;
     this.producto = p;
     this.next = null;
     this.prev = null;
   }

   public double getCantidad() { return this.cantidad; }
   public void setCantidad(double c) { this.cantidad = c; }

   public String getProducto() { return this.producto; }
   public void setProducto(String p) { this.producto = p; }

   public Term getNext() { return this.next; }
   public void setNext(Term t) { this.next = t; }

   public Term getPrev() { return this.prev; }
   public void setPrev(Term t) { this.prev = t; }
}
```

Modifique la gramática anterior para que el símbolo *Pedido* devuelva una lista de términos formada por objetos de la clase *Term*. Por ejemplo, para la entrada

3 * zumo + 2 * (café + tostada + 2 * porción)

la lista de términos a generar es la siguiente:



EJERCICIO 4 (2.5 puntos)

Considere la siguiente sintaxis LL(1) para las expresiones condicionales:

Condición → *CondiciónAnd Disyunción*

Disyuncion → "||" *CondiciónAnd Disyunción*

Disyunción → λ

CondiciónAnd → *CondiciónBase Conjunción*

Conjunción → "**&&**" *CondiciónBase Conjunción*

Conjunción → λ

CondiciónBase → *Expresión Operador Expresión*

CondiciónBase → "(" *Condición* ")"

Operador → "= ="

Operador → "**!**="

Operador → ">"

Operador → "<"

Operador → ">="

Operador → "<="

Genere un ETDS que permita obtener el código intermedio asociado a una condición teniendo en cuenta las siguientes consideraciones:

(a) Se dispone del método *getNewLabel()* para generar nuevas etiquetas.

(b) El símbolo *Expresión* tiene asociados dos atributos sintetizados: *code*, con el código intermedio que genera el valor de la expresión; y *temp*, con la referencia a la variable donde se almacena el valor de la expresión.

(c) El símbolo *Condición* debe tener asociados los siguientes atributos: *code*, atributo sintetizado con el código intermedio asociado a la condición; *label_true*, atributo heredado con el nombre de la etiqueta a la que debe saltar el código de la condición en caso de que ésta sea cierta; *label_false*, atributo heredado con el nombre de la etiqueta a la que debe saltar el código de la condición en caso de que ésta sea falsa.