Curso 2020/2021

Profesor de Teoría
SALVADOR VILLENA

PROCESADORES DE LENGUAJES

Relación de Problemas Nº 1

Procesadores de Lenguajes

1. Indicar en qué fase del proceso de compilación se realizaría cada una de las siguientes funciones:

   - Asignar una variable al registro 5.
   - Identificar `loop` como una etiqueta.
   - Detectar que en una sentencia de asignación los tipos no son compatibles.
   - Cambiar `ind + 4 * 4` por `ind + 16`.
   - Cambiar `ind + 16` por `Add #16, ind`.
   - Crear el árbol sintáctico.
   - Detectar que una variable no ha sido declarada.

---

2. Supongamos que deseamos realizar un compilador de un lenguaje L para una máquina M. En dicha máquina existe un compilador para el lenguaje Y que lo traduce a lenguaje ensamblador de M. El compilador para L se podría realizar en dos etapas:

   1: Usando el lenguaje Y escribir un compilador que traduce un pequeño subconjunto L_0 del lenguaje L a ensamblador M.

   2: Escribir un compilador en L_0 que traduzca L a lenguaje ensamblador M.

   Mostrar como usando estos programas podemos obtener un compilador para el lenguaje L, escrito en el ensamblador M y que lo traduce a ensamblador M.

---

3. Describa las siguientes sentencias utilizando notación BNF:

   - Sentencia `CASE` de Pascal.
   - Sentencia `IF` de ADA (igual que en Pascal salvo que ELSE es `ELSIF` y finaliza con un `ENDIF`).

---

4. Determinar cuáles de las siguientes funciones pueden ser llevadas a cabo por un procesador de lenguajes. En caso de que la respuesta sea afirmativa, en qué fase del análisis se basa primordialmente:

   (a) Dado un texto, realizar búsqueda y sustitución de texto.

   (b) Un generador de informes.

   (c) Implementar las operaciones de un expendedor de billetes de avión.

   (d) Dada la imagen de una pieza a tamaño real, comprobar si sus medidas se corresponden con unas preestablecidas.

---

5. ¿Es correcta la siguiente regla de especificación BNF de la sentencia while, de acuerdo a la especificación del lenguaje? ¿Hay cosas que no permite y debiera hacerlo?

```
<bloque> ::= BEGIN [<variables>] [<subprogramas>] {<sentencia>} END
<variables> ::= ...
<subprogramas> ::= ...
<sentencia> ::= <bloque> | <sentencia_while> | <asignacion> | ...
<sentencia_while> ::= WHILE ( <expresión> ) DO <bloque>
```

PROCESADORES DE LENGUAJES

Relación de Problemas Nº 2

Análisis de Léxico

1. Identificar los lexemas y los tokens que el analizador de léxico reconocería en los siguientes programas. Dar a cada token atributos razonables.

   (a) Lenguaje C:

```c
int max(int i, intj)
{
   /* devuelve el valor máximo de dos enteros */
   return i>j?i:j;
}
```

   (b) Fortran 77:

```
      FUNCTION MAX (I,J)
C     DEVUELVE EL MAXIMO DE LOS ENTEROS I Y J
      IF(I.GT.J) THEN MAX=I
      ELSE MAX=J
      END IF
      RETURN
```

   (c) Pascal:

```pascal
function max (i,j: integer):integer;
{Devuelve el valor máximo de dos enteros}
begin
   if i>j then max:=i
   else max := j
end;
```

---

2. Obtener la expresión regular para representar un número real (sin signo y sin el exponente) en Pascal. Obtenga el autómata finito determinista.

---

3. Construya una especificación LEX, para reconocer los siguientes tokens:

   (a) Identificador.

   (b) Cadena de caracteres formada por una secuencia de caracteres entre comillas.

   (c) $*$

   (d) $**$

   (e) $=$

   (f) $:=$

   (g) $+$

   (h) $-$

   (i) $/$

---

4. Dada la gramática siguiente:

$$
\begin{aligned}
S &\rightarrow \textbf{repeat}\ \text{exp}\ S\ \textbf{;} \\
&\mid \textbf{while}\ \text{exp}\ S\ \textbf{;} \\
&\mid \textbf{if}\ \text{exp}\ S\ \textbf{;} \\
&\mid \varepsilon \\
\text{exp} &\rightarrow \text{exp}\ \textbf{*}\ \text{exp} \\
&\mid \text{exp}\ /\ \text{exp} \\
&\mid \text{exp}\ +\ \text{exp} \\
&\mid \textbf{++}\ \text{exp} \\
&\mid \textbf{*}\ \text{exp} \\
&\mid \textbf{(}\text{exp}\textbf{)} \\
&\mid \text{const} \\
\text{const} &\rightarrow \text{const dig} \\
&\mid \text{dig} \\
\text{dig} &\rightarrow \textbf{0} \mid \textbf{1} \mid \cdots \mid \textbf{9}
\end{aligned}
$$

Obtener:

(a) El conjunto de tokens con el máximo nivel de abstracción suponiendo que se va a hacer traducción.

(b) El conjunto de tokens con el máximo nivel de abstracción suponiendo que sólo se va a hacer análisis sintáctico.

(c) Obtener la gramática abstracta para las dos situaciones anteriores.

---

5. Dada la construcción de un traductor en su fase de análisis léxico, indique la tabla de tokens con el máximo nivel de abstracción teniendo en cuenta únicamente los siguientes lexemas del lenguaje:

```
+, -, *, /, >, <, and, not, integer,
end, if, float, begin, char, true, false, j
```

---

6. Dada la siguiente gramática:

$$
\begin{aligned}
\text{Sent} &\rightarrow \textbf{repeat}\ \text{Exp Sent} \\
&\mid \textbf{while}\ \text{Exp Sent} \\
&\mid \textbf{if}\ \text{Exp Sent} \\
&\mid \varepsilon \\
\text{Exp} &\rightarrow \text{Const}\ \textbf{!=}\ \text{Exp} \\
&\mid \text{Const} < \text{Exp} \\
&\mid \text{Const} > \text{Exp} \\
&\mid \text{Const} + \text{Exp} \\
&\mid \textbf{*}\ \text{Exp} \\
&\mid +\ \text{Exp} \\
&\mid \text{Const} \\
\text{Const} &\rightarrow \text{Const dig} \\
&\mid \text{num} \\
\text{dig} &\rightarrow \textbf{0} \mid \textbf{1} \mid \cdots \mid \textbf{9}
\end{aligned}
$$

Obtener la tabla de tokens con el máximo nivel de abstracción suponiendo que:

(a) Se va a realizar traducción.

(b) Sólo se va a realizar análisis sintáctico.

---

7. Sea la gramática con las producciones siguientes:

$$
\begin{aligned}
\text{Exp} &\rightarrow \text{Exp} + \text{Exp} \\
&\mid \text{Exp}\ \textbf{**}\ \text{Exp} \\
&\mid \textbf{**}\ \text{Exp} \\
&\mid +\ \text{Exp} \\
&\mid \textbf{(}\ \text{Term} \\
&\mid \text{Num} \\
\text{Term} &\rightarrow \text{Exp} \\
&\mid \textbf{)} \\
\text{Num} &\rightarrow \text{Num Dig} \\
&\mid \text{num} \\
\text{Dig} &\rightarrow \textbf{0} \mid \textbf{1} \mid \cdots \mid \textbf{9}
\end{aligned}
$$

Obtener la tabla de tokens con el máximo nivel de abstracción suponiendo que:

(a) Se va a realizar traducción.

(b) Sólo se va a realizar análisis sintáctico.

---

8. Sea la gramática con las producciones siguientes:

$$
\begin{aligned}
\text{programa} &\rightarrow \textbf{begin}\ \text{sentencias}\ \textbf{end} \\
\text{sentencias} &\rightarrow \text{sentencias}\ \textbf{;}\ \text{sentencia} \\
&\mid \text{sentencia} \\
\text{sentencia} &\rightarrow \text{ident}\ \textbf{=}\ \text{exp} \\
&\mid \text{programa} \\
\text{exp} &\rightarrow \text{exp}\ \textbf{+}\ \text{exp} \\
&\mid \text{exp}\ \textbf{-}\ \text{exp} \\
&\mid \textbf{-}\ \text{exp} \\
&\mid \text{const} \\
&\mid \text{ident} \\
\text{const} &\rightarrow \text{const num} \\
&\mid \text{num} \\
\text{num} &\rightarrow \textbf{0} \mid \textbf{1} \mid \cdots \mid \textbf{9} \\
\text{ident} &\rightarrow \text{ident letra} \\
&\mid \text{letra} \\
\text{letra} &\rightarrow \textbf{a} \mid \textbf{b} \mid \cdots \mid \textbf{z}
\end{aligned}
$$

Obtener la tabla de tokens con el máximo nivel de abstracción suponiendo que:

(a) Se va a realizar traducción.

(b) Sólo se va a realizar análisis sintáctico.

---

9. Sea la gramática con las producciones siguientes:

$$
\begin{aligned}
\text{programa} &\rightarrow \text{funciones main} \\
&\mid \text{main} \\
\text{funciones} &\rightarrow \text{funciones funcion} \\
&\mid \text{funcion} \\
\text{funcion} &\rightarrow \textbf{function}\ \text{ident}\ \textbf{(}\ \textbf{)}\ \text{bloque} \\
\text{main} &\rightarrow \textbf{main}\ \text{bloque} \\
\text{bloque} &\rightarrow \textbf{\{}\ \text{sentencias}\ \textbf{\}} \\
\text{sentencias} &\rightarrow \text{sentencias sentencia} \\
&\mid \text{sentencia} \\
\text{sentencia} &\rightarrow \text{ident}\ \textbf{=}\ \text{expresion}\ \textbf{;} \\
&\mid \text{bloque} \\
\text{expresion} &\rightarrow \text{expresion}\ \textbf{*}\ \text{expresion}\ \textbf{**}\ \text{expresion} \\
&\mid \text{expresion}\ \textbf{\&}\ \text{expresion} \\
&\mid \textbf{*}\ \text{expresion} \\
&\mid \textbf{\&}\ \text{expresion} \\
&\mid \text{ident} \\
&\mid \text{ident}\ \textbf{(}\ \textbf{)} \\
&\mid \text{const} \\
\text{const} &\rightarrow \text{num.num} \\
&\mid \text{num} \\
\text{num} &\rightarrow \textbf{0} \mid \textbf{1} \mid \cdots \mid \textbf{9} \\
\text{ident} &\rightarrow \text{ident letra} \\
&\mid \text{letra} \\
\text{letra} &\rightarrow \textbf{a} \mid \textbf{b} \mid \cdots \mid \textbf{z}
\end{aligned}
$$

Obtener la tabla de tokens con el máximo nivel de abstracción suponiendo que:

(a) Se va a realizar traducción.

(b) Sólo se va a realizar análisis sintáctico.

---

10. Dada la siguiente especificación léxica en sintaxis de LEX ¿qué problema plantea a la hora de reconocer los tokens? ¿De qué forma se podría corregir?

```
.......
%%
.....
```

```
[a-zA-Z]([a-zA-Z0-9])*  { return ID; }
"while"                 { return WHILE; }
"if"                    { return IF; }
.....
%%
.....
```

PROCESADORES DE LENGUAJES

Relación de Problemas Nº 3

Análisis Sintáctico

1. Demostrar que la gramática formada por las siguientes producciones:

$$S \rightarrow aSbS \mid bSaS \mid \varepsilon$$

es ambigua construyendo el árbol sintáctico para la cadena de entrada `abab`.

---

2. Diseñar una gramática no ambigua que genere el lenguaje de todos los números enteros sin signo que sean pares, considerando para ello que el 0 es par. Un ejemplo de números que debe generar la gramática sería: 0, 234, 11112, 00078.

---

3. Demostrar que la gramática formada por las siguientes producciones:

$$\begin{aligned} \text{Numero} &\rightarrow \text{Digito} \\ &\mid \text{Numero Numero} \\ \text{Digito} &\rightarrow 0 \mid 1 \mid \cdots \mid 9 \end{aligned}$$

es ambigua. Obtener una gramática equivalente que no lo sea.

---

4. Dada la siguiente gramática:

$$\begin{aligned} S &\rightarrow (\; S * S \;) \\ &\mid \; x \end{aligned}$$

   (a) Construya el árbol sintáctico para las siguientes sentencias: `x`, `(x*(x*x))`, `x*x`.

   (b) Obtener las sentencias anteriores aplicando derivaciones más a la derecha y más a la izquierda.

---

5. Dada la siguiente gramática:

$$\begin{aligned} S &\rightarrow x \\ &\mid \; (\; S \; R \\ R &\rightarrow , \; S \; R \\ &\mid \;) \end{aligned}$$

   (a) Demostrar que es una gramática LL(1).

   (b) Crear la tabla de análisis LL(1).

   (c) Mostrar paso a paso el análisis de la sentencia de entrada: `(x,(x,x))`.

---

6. Comprobar si las gramáticas siguientes son LL(1):

(a)
$$\begin{aligned} S &\rightarrow aABC \\ A &\rightarrow a \mid bbD \\ B &\rightarrow a \mid \varepsilon \\ C &\rightarrow b \mid \varepsilon \\ D &\rightarrow c \mid \varepsilon \end{aligned}$$

(b)
$$\begin{aligned} S &\rightarrow AD \\ A &\rightarrow bB \mid \varepsilon \\ B &\rightarrow Ca \mid D \\ C &\rightarrow a \mid \varepsilon \\ D &\rightarrow b \mid c \end{aligned}$$

(c)
$$\begin{aligned} S &\rightarrow BDf \mid e \\ B &\rightarrow eD \mid aB \\ D &\rightarrow d \mid \varepsilon \end{aligned}$$

(d)
$$\begin{aligned} D &\rightarrow bA \mid cX \mid d \\ X &\rightarrow MbA \mid BF \\ M &\rightarrow cM \mid \varepsilon \\ A &\rightarrow MBj \mid F \\ B &\rightarrow c \mid cBh \mid \varepsilon \\ F &\rightarrow fA \mid \varepsilon \end{aligned}$$

---

7. La siguiente gramática abstrae la sentencia IF de acuerdo a las siguientes producciones:

$$\begin{aligned} S &\rightarrow iEtSS' \mid a \\ S' &\rightarrow eS \mid \varepsilon \\ E &\rightarrow c \end{aligned}$$

donde i=IF, e=ELSE, a=sentencia, t=THEN y c=CONDICION.

(a) Demostrar que no es LL(1).

(b) Crear la tabla de análisis.

(c) Muestre cómo se analizaría paso a paso la sentencia de entrada siguiente:
`IF c THEN IF c THEN a ELSE a`.

(d) Modifique la tabla de análisis para que cada ELSE se asocie con el IF más cercano.

---

8. Dadas la siguiente gramática:

$$\begin{aligned} E &\rightarrow TE' \\ E' &\rightarrow +TE' \mid \varepsilon \\ T &\rightarrow FT' \\ T' &\rightarrow *FT' \mid \varepsilon \\ F &\rightarrow id \mid (E) \end{aligned}$$

(a) Construir la tabla de análisis LL(1).

(b) Analizar cada entrada de error e indicar el mensaje apropiado y posibles acciones a adoptar sobre la pila y la entrada para poder recuperarse.

---

9. Dada la siguiente gramática:

$$H \rightarrow Hab \mid Hjc \mid mHj \mid ck \mid tHj \mid \varepsilon$$

(a) Eliminar la recursividad a la izquierda.

(b) ¿Es la gramática resultante LL(1)?

---

10. Factorizar las siguientes gramáticas y comprobar si las gramáticas resultantes son LL(1):

(a)
$$\begin{aligned} Z &\rightarrow bNAc \mid bNc \mid bAj \mid bNA \mid cN \mid j \\ N &\rightarrow x \\ A &\rightarrow h \mid z \end{aligned}$$

(b)
$$\begin{aligned} S &\rightarrow wAz \mid xBz \mid wBy \mid wBz \mid xAy \\ A &\rightarrow v \\ B &\rightarrow v \end{aligned}$$

(c)
$$\begin{aligned} Z &\rightarrow bNAc \mid Nc \mid bj \mid cA \mid t \\ N &\rightarrow bk \mid cH \\ A &\rightarrow hN \mid z \\ H &\rightarrow n \mid k \end{aligned}$$

---

11. A partir de una gramática para generar expresiones lógicas:

$$\begin{aligned} E &\rightarrow C \mid E \text{ or } C \\ C &\rightarrow I \mid C \text{ and } I \\ I &\rightarrow P \mid \text{not } P \\ P &\rightarrow t \mid f \mid (E) \end{aligned}$$

Identificar las relaciones de precedencia simple que se establecería en la pila del autómata para las siguientes formas de sentencia analizadas:

(a) `t and f or t`

(b) `C or I`

(c) `E or I`

(d) `E or C`

(e) `E or C and I`

(f) `C and I or t`

(g) `f and not t`

---

12. Sean las gramáticas:

(a)
$$\begin{array}{l} E \to C \mid E \text{ or } C \\ C \to I \mid C \text{ and } I \\ I \to P \mid \text{not } P \\ P \to t \mid f \mid (E) \end{array}$$

(b)
$$\begin{array}{l} E \to E{+}T \mid T \\ T \to TF \mid F \\ F \to F{*} \mid a \mid b \end{array}$$

(c)
$$\begin{array}{l} S \to aAd \mid bec \mid bAc \\ A \to e \end{array}$$

(d)
$$\begin{array}{l} S \to AD \\ A \to bB \mid \varepsilon \\ B \to Ca \mid D \\ C \to a \mid \varepsilon \\ D \to b \mid c \end{array}$$

(e)
$$\begin{array}{l} L \to Lb \mid aAC \\ A \to dA \mid b \\ C \to Ad \mid f \end{array}$$

(a) Comprobar si son gramáticas de precedencia de operador.

(b) Construir la tabla de precedencia simple.

---

13. Calcular la tabla de análisis SLR, LR(1) y LALR para las siguientes gramáticas:

(a)
$$\begin{array}{l} S \to Aa \mid bAc \mid Bc \mid bBa \\ A \to d \\ B \to d \end{array}$$

(b)
$$\begin{array}{l} S \to wAz \mid xBz \mid wBy \mid wBz \mid xAy \\ A \to v \\ B \to v \end{array}$$

(c)
$$\begin{array}{l} S \to aSD \mid b \\ D \to Dc \mid e \end{array}$$

(d)
$$\begin{array}{l} S \to aDS \mid b \\ D \to cD \mid \varepsilon \end{array}$$

(e)
$$\begin{array}{l} S \to AD \\ A \to mD \mid \varepsilon \\ D \to b \mid c \end{array}$$

(f)
$$\begin{array}{l} S \to ADk \\ A \to Df \mid \varepsilon \\ D \to b \mid c \end{array}$$

(g)
$$\begin{array}{l} S \to badD \\ D \to Df \mid a \mid \varepsilon \end{array}$$

(h) $S \to aSbb \mid a \mid \varepsilon$

---

14. Sea la gramática:

$$\begin{array}{l} S \to XX \\ X \to aX \mid b \end{array}$$

Construir su tabla de análisis SLR y estudiar todas las entradas de error indicando cuál es el mensaje de error adecuado y las acciones a realizar sobre la pila y la entrada para intentar recuperarse.

---

15. La siguiente gramática:

$$\begin{array}{l} E \to E + E \\ \quad \mid E * E \\ \quad \mid E / E \\ \quad \mid E - E \\ \quad \mid (E) \\ \quad \mid id \end{array}$$

es ambigua. Construir su tabla de análisis SLR y analizar los conflictos que aparecen debido a la ambigüedad. Estudiar cómo se podrían eliminar (como sugerencia, establecer relaciones de precedencia entre los operadores, por ejemplo, dar a los operadores de multiplicar y dividir mayor prioridad que a la suma y la resta y que todos sean asociativos a la izquierda).

---

16. Dada la siguiente gramática:

$$\begin{array}{l} S \to A\ B \\ A \to \textbf{begin}\ S\ \textbf{end}\ B\ \textbf{theend} \\ \quad \mid \varepsilon \\ B \to \textbf{var}\ L : \text{tipo} \\ \quad \mid B\ \textbf{endvar} \\ \quad \mid \varepsilon \\ L \to L, id \\ \quad \mid id \end{array}$$

(a) Demostrar que esta gramática no es LL(1). Hacer las transformaciones necesarias para que lo sea.

(b) Construir la tabla de análisis LL(1) para la nueva gramática obtenida en el apartado anterior.

(c) Analizar la siguiente entrada y determinar si es o no correcta sintácticamente:

```
begin
   var a, b : tipo
   var c : tipo
endvar
end
var d : tipo
theend
```

---

17. La siguiente gramática abstrae la sentencia IF de acuerdo a las siguientes producciones:

$$\begin{aligned} S &\rightarrow iEtSS' \mid a \\ S' &\rightarrow eS \mid \varepsilon \\ E &\rightarrow c \end{aligned}$$

donde i=IF, e=ELSE, a=sentencia, t=THEN y c=CONDICION.

(a) Demostrar que no es SLR, LR(1) ni LALR(1).

(b) Modifique la tabla de análisis para que cada ELSE se asocie con el IF más cercano.

---

18. Dada la gramática que genera expresiones aritméticas con los operadores de suma y resta de acuerdo a las siguientes producciones:

$$\begin{aligned} (1)\ E &\rightarrow E + E \\ (2) \quad &\mid E - E \\ (3) \quad &\mid id \end{aligned}$$

Cuya tabla de análisis SLR es la siguiente:

	i	+	-	$	E
0	d2				1
1		d3	d4	acc	
2		r3	r3	r3	
3	d2				5
4	d2				6
5		d3/r1	d4/r1	r1	
6		d3/r2	d4/r2	r2	

¿Cuáles serían las entradas sin conflictos para los estados 5 y 6 sin contemplamos las siguientes suposiciones en cuanto a precedencia y asociatividad entre los operadores?

(a) + asociativo a la izquierda, - asociativo a la derecha y - tiene más precedencia que el +.

(b) + asociativo a la derecha, - asociativo a la derecha y - tiene más precedencia que el +.

(c) + asociativo a la izquierda, - asociativo a la derecha y + tiene más precedencia que el -.

(d) + asociativo a la derecha, - asociativo a la derecha y + tiene más precedencia que el -.

(e) + asociativo a la izquierda, - asociativo a la izquierda y - tiene más precedencia que el +.

(f) + asociativo a la izquierda, - asociativo a la izquierda y - tiene más precedencia que el +.

(g) ambos son asociativos a la izquierda y tienen la misma precedencia.

(h) ambos son asociativos a la derecha, y tienen la misma precedencia.

---

19. Dada la siguiente especificación en sintaxis de YACC:

```
%token TOKEN_IF TOKEN_THEN TOKEN_ELSE
......
%%
......
sentencia_if : TOKEN_IF expresion TOKEN_THEN sentencia
             | TOKEN_IF expresion TOKEN_THEN sentencia TOKEN_ELSE sentencia ;
......
%%
......
```

(a) ¿Qué tipo de conflicto provoca esta producción?

(b) ¿Podría eliminarse dicho conflicto? ¿cómo?

(c) En caso de ignorar dicho conflicto, ¿actuaría el analizador sintáctico de forma correcta? Razone la respuesta.

---

20. Dada la siguiente especificación obtenida del archivo `y.output` que produce el YACC a la hora de generar el analizador sintáctico:

```
  0  $accept : programa $end

  1  programa : PRINCIPAL bloque

  2  bloque : INICIOBLOQUE sentencias FINBLOQUE

  3  sentencias : sentencias sentencia PCOMA
  4             | sentencia PCOMA

  5  sentencia : bloque
  6            | sentencia_asig
  7            | subprograma
  8            | expresion

  9  subprograma : IDENTIFICADOR ABREPARENT lista_expr CIERRAPARENT

 10  lista_expr : lista_expr COMA expresion
 11             | expresion

 12  sentencia_asig : IDENTIFICADOR ASIGNACION expresion

 13  expresion : ABREPARENT expresion CIERRAPARENT
 14                | expresion OPUNABIN expresion
 15                | expresion OPBIN expresion
 16                | OPUNABIN expresion
 17                | IDENTIFICADOR
 18                | CONSTANTE
```

Indicar las producciones de error necesarias para que el analizador pueda recuperarse ante las siguientes situaciones:

(a) Error en `expresion`.

(b) Error en el token `COMA`

(c) Error en el token `PCOMA`

PROCESADORES DE LENGUAJES

Relación de Problemas Nº 4

Análisis Semántico

1. Sea la siguiente gramática que genera cadenas de 1 a *n* veces el carácter ”a”:

$$L \rightarrow AL \mid A$$
$$A \rightarrow \mathbf{a}$$

Obtener la gramática con atributos que, al analizar una cadena, calcule el número de veces que aparece el mencionado carácter.

---

2. Considerar la siguiente gramática que genera números binarios:

$$S \rightarrow L.L \mid L$$
$$L \rightarrow LB \mid B$$
$$B \rightarrow \mathbf{0} \mid \mathbf{1}$$

Obtener la gramática con atributos que, al analizar un número binario, calcule su equivalente en decimal.

---

3. La siguiente gramática reproduce la estructura de una sentencia `for`:

$$\begin{aligned} S &\rightarrow \textbf{for}\ \text{id=E}\ \textbf{to}\ \text{E}\ \textbf{step}\ \text{E}\ \textbf{do}\ \text{S} \\ &\mid\ \text{S}\ \textbf{;}\ \text{S} \\ &\mid\ \text{o} \\ E &\rightarrow \text{E op\_aritmetico E} \\ &\mid\ \text{E op\_logico E} \\ &\mid\ \text{num.num} \\ &\mid\ \text{num} \\ &\mid\ \textbf{true} \\ &\mid\ \textbf{false} \\ &\mid\ \textbf{id} \end{aligned}$$

Obtener la gramática con atributos para realizar las comprobaciones semánticas necesarias. Indique qué información sería necesaria tener previamente almacenada en una tabla de símbolos y las funciones que se necesiten para su implementación. Hay que tener en cuenta que el lenguaje no admite coherciones.

---

4. Sea la siguiente gramática que genera sentencias de asignación:

$$S \rightarrow \textbf{id :=}\ E$$
$$E \rightarrow E\textbf{+}T \mid T$$
$$T \rightarrow T\textbf{*}F \mid F$$
$$F \rightarrow \textbf{(}E\textbf{)} \mid \textbf{id} \mid \text{num.num} \mid \text{num}$$

Obtener la gramática con atributos que realice la comprobación de tipos en los siguientes supuestos:

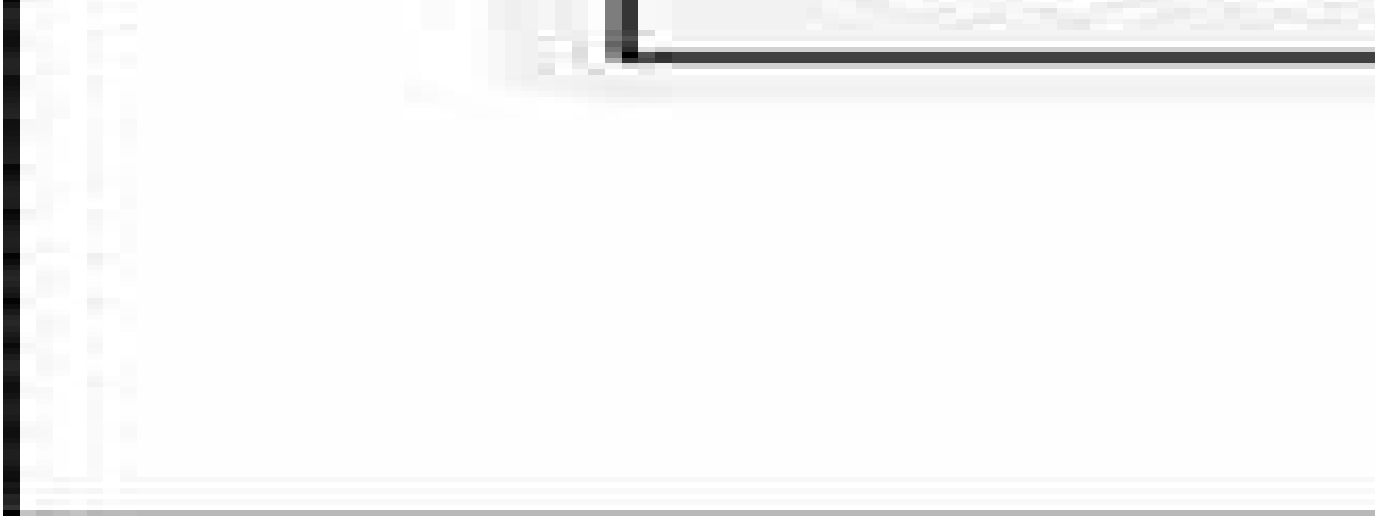
(a) El lenguaje no admite coherciones.

(b) El lenguaje admite coherciones. En concreto, se permite el paso de entero a real.

En ambos casos, se debe suponer la existencia de una tabla de símbolos con funciones de acceso a la misma para determinar el tipo de las variables que en ella estén alojadas.

---

5. Se desea realizar una auditoría de los libros de contabilidad de una empresa. En los libros de contabilidad aparece la información distribuida según el siguiente formato:



Donde:

- **Línea de principio de página:** Está compuesta por la siguiente información:
  - *Texto*: ”Suma anterior:”.
  - *Haber*: Número con 8 cifras enteras y 2 decimales.
  - *Debe* : Número con 8 cifras enteras y 2 decimales.
- **Línea de asientos contables:**
  - *Número de asiento*: Entero de 5 dígitos.
  - *Fecha*: Composición dia/mes/año, todos con 2 cifras enteras cada uno.
  - *Código de cuenta contable*: Compuesto por la siguiente información:
    - Código cuenta primaria: 1 dígito.
    - Código cuenta secundaria: 2 dígitos.
    - Código cuenta terciaria: 6 dígitos.
  - *Descripción*: Texto indicativo de la operación, basado en 45 caracteres alfabéticos.
  - *Haber*: Número con 8 cifras enteras y 2 decimales.
  - *Debe*: Número con 8 cifras enteras y 2 decimales.
- **Línea de final de página:**
  - *Texto*: ”Suma y sigue:”.
  - *Haber*: Número con 8 cifras enteras y 2 decimales.
  - *Debe*: Número con 8 cifras enteras y 2 decimales.

Entre una línea y otra puede aparecer una o más marcas de final de línea y entre campos en cada línea aparecen uno o más espacios en blanco. Ambas situaciones se deben considerar ignoradas a nivel léxico.

Deseamos realizar comprobaciones sobre:

- Enumeración de las **líneas de asientos**.
- La suma del debe y haber de las líneas de asientos debe ser concordante con el debe y haber de las líneas de principio y final de página.
- Las líneas de asiento deben estar ordenadas cronológicamente.
- Determinar el número de asientos contables que sean incorrecto por error u omisión de información.
- Obtener el total del debe y haber de aquellas líneas de asientos formadas por las cuentas cuyo código de cuenta primaria sea 6 (por ejemplo, 620000001, 601011005, ...)

(a) Defina el lenguaje de los libros de contabilidad mediante una gramática con las producciones en forma gramatical tal que, si se detecta un error sintáctico en alguno de los campos que componen una línea contable, se recupere y continúe salvo que el mismo suceda en los campos *Haber*, *Debe* y *Código de cuenta contable*, en cuyo caso debemos descartar la línea de asiento contable al completo.

(b) Definir los tokens con el máximo nivel de abstracción para contemplar las comprobaciones anteriormente mencionadas y obtenga la gramática abstracta correspondiente.

(c) Definir los tokens con el máximo nivel de abstracción en el caso de que sólo se requera realizar análisis sintáctico.

(d) Definir la gramática con atributos en sintaxis de YACC a partir de la obtenida en el apartado (b) para que contemple las comprobaciones que se exigen en el problema.

---

6. Considerar un archivo cuyos registros constan de los siguientes campos:

Código	Descripción	Cantidad	Precio unitario	Precio total
8 caracteres alfanuméricos	Cadena alfanumérica	Número entero	Número real	Número real

Se desean realizar las siguientes comprobaciones:

- Que en cada registro el valor del campo **Precio total** sea igual al producto del campo **Cantidad** y **Precio unitario**.
- Obtener el importe total y el número de productos vendidos.
- Si el campo **Código**, **Cantidad** o **Precio unitario** es incorrecto a nivel sintáctico, se deberá avisar de registro erróneo y saltar a otro. Si hubiere un error sintáctico en cualquiera de los campos restantes, debemos avisar, recuperarnos y continuar el análisis sin perder el resto de la información existente en dicho registro.

(a) Definir una gramática que genere el texto correspondiente a ese archivo y clasificarla según la jerarquía de Chomsky.

(b) Definir los tokens con el máximo nivel de abstracción y que se adecue a los objetivos anteriores.

(c) Definir los tokens con el máximo nivel de abstracción en el caso de que sólo se requiera realizar análisis sintáctico.

(d) A partir de la gramática abstracta obtenida en el apartado (b), especificar en YACC una gramática con atributos que realice las comprobaciones que se citan en el encabezamiento del problema.

---

7. Dada la lista de llamadas emitidas y recibidas por parte de una empresa con un formato similar al que muestra la siguiente tabla:

Nº Teléfono	Tipo	Fecha	Hora	Duración	Operador
658242424	R	02/12/2005	09:02	45	V
952343434	E	02/12/2005	10:03	33	
.....					
955346474	R	03/12/2005	12:56	123	

Donde los campos expresan la siguiente información:

- **Nº Teléfono**: 9 dígitos correspondientes al número de teléfono.
- **Tipo**: Representa las dos modalidades: **R**ecibida o **E**mitida.
- **Hora**: Formato estándar de hora/minuto (dos dígitos cada uno) en la que se inició la llamada.
- **Duración**: Formato en segundos que indica la duración de la llamada.
- **Operador**: Indicativo cuando la llamada origen procede de un teléfono móvil, en cuyo caso puede ser de tres valores (**M/V/A**). En caso de ser un teléfono fijo, dicho campo estará vacío.

Se desean realizar las siguientes comprobaciones:

- Para cada tipo de Operador, obtener el número de móviles distintos que han intervenido en la lista de llamadas controladas.
- Obtener el tiempo total en llamadas emitidas a móviles y el tiempo total en llamadas emitidas a números fijos.
- Obtener el tiempo total en llamadas recibidas.

(a) Definir una gramática que genere el texto correspondiente a ese archivo y clasificarla según la jerarquía de Chomsky.

(b) Definir los tokens con el máximo nivel de abstracción y que se adecue a los objetivos planteados en el problema.

(c) Especificar una gramática con atributos en sintaxis de YACC que realice las comprobaciones que se exponen en el encabezado del problema.

---

8. En uno de los archivos "log"de un firewall de un sistema operativo linux nos informa de una serie de accesos al sistema. Unos de ellos son permitidos y otros no de acuerdo a la configuración del firewall. El formato de dicho archivo responde a la siguiente gramática en sintaxis de YACC:

```
%token IPSOURCE HOSTNAME MESSAGE ACCESS
%token MONTH DAY YEAR HOUR MIN
%%
archivo_log : archivo_log datos_log
            | datos_log ;
datos_log   : IPSOURCE fechahora HOSTNAME MESSAGE ACCESS ;
fechahora   : MONTH DAY YEAR HOUR MIN ;
%%
```

Donde:

- `HOSTNAME`: Nombre del host donde se realiza el intento de conexión.
- `MESSAGE`: Mensaje de texto indicando la acción.
- `IPSOURCE`: Dirección IP desde donde se intenta acceder.
- `ACCESS`: Dispone de dos valores `ACCEPT` o `DENY`. El primero indica que se ha permitido el acceso y el segundo indica que dicho intento de acceso ha sido rechazado.

(a) Modificar la gramática inicial para que cuando se produzca un error en cualquiera de los valores de `fechahora`, `HOSTNAME` y `MESSAGE`, se pueda recuperar ante el error.

(b) Dada la gramática obtenida en el apartado (a), añadir las acciones semánticas necesarias para determinar cuántas entradas del archivo "log"han sido detectadas como erróneas.

(c) Dada la gramática obtenida en el apartado (a), añadir las acciones semánticas necesarias para determinar la lista de direcciones IP diferentes que aparecen en el "log$z$ el número de veces que han intentado acceder al sistema con respuesta de acceso denegada.

---

9. Dado un sistema de control de acceso de los trabajadores de una empresa a una determinada zona restringida que contabiliza los accesos de entrada y de salida, se pretende realizar una serie de acciones complementarias al ”log” del sistema ilustrado en la siguiente tabla:

CODE	PIN	DAY	MONTH	YEAR	HOUR	MINUTE	ACCESS
				.....			
				.....			
34219883	3445	21	06	05	17	01	IN
43300944	0909	21	06	05	17	02	IN
				.....			
				.....			
34219883	3445	21	06	05	17	22	OUT
43300944	0909	21	06	05	17	45	OUT
				.....			
				.....			

Dichos resultados se pueden expresar mediante la siguiente gramática:

$$
\begin{aligned}
\text{system\_log} &\rightarrow \text{system\_log line\_log} \\
&\mid \text{line\_log} \\
\text{line\_log} &\rightarrow \text{CODE PIN fecha access} \\
\text{fecha} &\rightarrow \text{DAY MONTH YEAR HOUR MINUTE} \\
\text{access} &\rightarrow \text{IN} \\
&\mid \text{OUT}
\end{aligned}
$$

Donde:

- CODE: 8 caracteres alfanuméricos.
- PIN: 4 dígitos.
- DAY, MONTH, YEAR, HOUR, MINUTE: 2 dígitos.
- IN: secuencia de caracteres ’IN’ indicativos de que se ha realizado una entrada en zona restringida.
- OUT: secuencia de caracteres ’OUT’ indicativos de que se ha realizado una salida de la zona restringida.

Se debe tener en cuenta que cuando un trabajador realiza un acceso de entrada, después realiza forzosamente un acceso de salida en un periodo de tiempo no definido.

Especificar una gramática con atributos en sintaxis de YACC que realice las siguientes acciones:

(a) Determinar el tiempo de permanencia en zona restringida de cada uno de los trabajadores.

(b) Determinar el número de accesos (tanto de entrada como de salida) de cada uno de los trabajadores.

Todas las estructuras de datos y funciones de manejo de las mismas que se consideren necesarias para el desarrollo de los apartados no tienen que ser implementadas pero sí definidas. Igualmente se puede suponer la existencia de un conjunto de funciones de manejo de tiempos con objeto de calcular los tiempos de permanencia.

---

10. Deseamos comprobar el funcionamiento de una aplicación de caja de un supermercado. Las transacciones que se realizan en la caja obedecen a la siguiente gramática:

$$
\begin{aligned}
\mathrm{L} &\rightarrow \mathrm{L\ P} \\
&\mid \mathrm{P} \\
\mathrm{P} &\rightarrow \mathrm{V}\ \texttt{importetotal} \\
\mathrm{V} &\rightarrow \mathrm{V\ D} \\
&\mid \mathrm{D} \\
\mathrm{D} &\rightarrow \texttt{cantidad preventa precompra importe}
\end{aligned}
$$

Donde:

- P representa el ticket de venta de un cliente compuesto de un conjunto de líneas de producto V y el `importetotal`.
- V es el conjunto de líneas de productos.
- D es una línea de venta de productos.
- L representa todas las ventas de todos los clientes.
- `cantidad` es el número de unidades adquiridas de un cierto producto.
- `preventa` es el precio de venta al público por unidad.
- `precompra` es el precio de compra por parte del establecimiento por unidad.
- `importe` es el precio total de la línea de un producto.

Proponer una gramática con atributos en sintaxis de YACC que realice las siguientes comprobaciones:

(a) Comprobar que el `importe` de cada línea de producto sea el resultado de multiplicar `cantidad` y `preventa` y que `importetotal` sea la suma de todos los importes del ticket.

(b) A partir de una cantidad de dinero existente en la caja que se almacena en la variable `METALICO`, obtener la cantidad de dinero que debe existir en la caja al final de procesar todas las ventas.

(c) Obtener el beneficio neto después de procesar todas las ventas realizadas.

---

11. Considerar la siguiente gramática simplificada para referencias de arrays:

$$
\begin{aligned}
\mathrm{L} &\rightarrow \mathrm{id} \\
&\mid \mathrm{id}\ [\ \mathrm{E}\ ] \\
\mathrm{E} &\rightarrow \mathrm{E}\ ,\ \mathrm{R} \\
&\mid \mathrm{R} \\
\mathrm{R} &\rightarrow \mathrm{id}
\end{aligned}
$$

(a) Defina un atributo deniminado `numDim` y las reglas semánticas asociadas a cada producción de forma que al analizar una sentencia obtengamos el número de dimensiones que tiene el array que es referenciado.

(b) ¿De qué tipo es el atributo `numDim` empleado en el apartado anterior?

(c) Mostrar el árbol sintáctico y la evaluación de los atributos para la sentencia `A[i,j,k]` que representa la siguiente secuencia de tokens: `id [ id , id , id ]`.

(d) ¿Qué otra utilidad podría proporcionar la consideración de este atributo desde el punto de vista del análisis semántico?

PROCESADORES DE LENGUAJES

Relación de Problemas Nº 5

Lenguajes Intermedios y Generación de Código

1. Dada la siguiente sentencia: `A:= 2*(Bee+Cee/Dee)`. Obtener:
   (a) Una gramática razonable que la genere.
   (b) El árbol sintáctico de dicha sentencia.
   (c) El árbol abstracto.
   (d) Su traducción a cuartetos.
   (e) Su traducción a tercetos.
   (f) Su traducción a tercetos indirectos.

---

2. Repetir el problema anterior para la sentencia: `while (1<P) and (P<3) do P:= P + Q`

---

3. Sea la siguiente gramática que genera sentencias de asignación:

$$\begin{aligned} S &\rightarrow \text{id} := E \\ E &\rightarrow E+T \mid T \\ T &\rightarrow T*F \mid F \\ F &\rightarrow \text{id} \mid \text{num} \mid \text{num.num} \mid (E) \end{aligned}$$

   Obtener la gramática con atributos que realice las siguientes funciones:
   (a) Generar código intermedio utilizando cuartetos.
   (b) Generar código intermedio utilizando cuartetos pero, en este caso, debe eliminar la sobrecarga de los operadores * y + de tal forma que cuando la operación se realice entre enteros deberá traducirlos por `mul_ent` y `sum_ent` respectivamente y `mul_real` y `sum_real` cuando se realice entre reales.

---

4. La definición dirigida por sintaxis siguiente traduce la regla "E → id1 <id2" a:

```
            if id1<id2 goto E.verdadera
            goto E.falsa
```

   En lugar de ésto, se podría traducir por una sola proposición de la siguiente forma:

```
            if id1>=id2 goto E.falsa
E.verdadera:
```

   Obtener una gramática con atributos que realice dicha traducción.

---

5. Sea la gramática:

$$\begin{aligned} P &\rightarrow D \,;\, S \\ D &\rightarrow D \,;\, D \\ &\mid L \,;\, T \\ &\mid \texttt{type}\ \text{id} : T \\ L &\rightarrow L \,,\, \texttt{id} \mid \text{id} \\ T &\rightarrow \texttt{char} \mid \texttt{entero} \mid \texttt{real} \mid \texttt{booleano} \\ &\mid \texttt{array}\ [\text{num}]\ \texttt{of}\ T \mid \hat{}\,T \mid T \texttt{->} T \mid \text{id} \\ S &\rightarrow \text{id}\ \texttt{:=}\ E \\ &\mid \texttt{if}\ E\ \texttt{then}\ S \\ &\mid \texttt{while}\ E\ \texttt{do}\ S \\ &\mid S \,;\, S \\ E &\rightarrow \text{literal} \mid \text{num} \mid \text{id} \mid E[E] \mid E\hat{}\, \mid E\ \texttt{op\_logico}\ E \mid E \bmod E \mid E(E) \mid E\ \texttt{op\_arit}\ E \end{aligned}$$

   Realizar la asignación y comprobación de tipos cuando se utiliza:
   (a) Equivalencia nominal (por nombre).

(b) Equivalencia estructural.

---

6. La siguiente gramática define listas de listas de literales:

$$
\begin{aligned}
P &\rightarrow D\ ;\ E \\
D &\rightarrow D\ ;\ D \mid id : T \\
T &\rightarrow \texttt{lista de}\ T \mid \texttt{caracter} \mid \texttt{entero} \\
E &\rightarrow (L) \mid literal \mid num \mid id \mid \texttt{nil} \\
L &\rightarrow E\ ,\ L \mid E
\end{aligned}
$$

Donde D representa una secuencia de declaraciones, T es el tipo asignado a un identificador, E son las expresiones del programa formadas por listas, L es una secuencia de listas. El símbolo terminal `nil` indica que una expresión puede ser la lista vacía y, por tanto, es una lista de elementos de cualquier tipo.

Escribir las reglas de traducción para asignar tipo a las expresiones E y las listas L.

---

7. Las siguientes producciones definen la sintaxis de una sentencia `for` para el lenguaje Pascal:

$$
\begin{aligned}
S &\rightarrow \texttt{for}\ id\ \texttt{:=}\ exp\ \texttt{to}\ exp\ I\ \texttt{do}\ S \\
I &\rightarrow \texttt{inc}\ exp \mid \texttt{decr}\ exp \mid \varepsilon \\
S &\rightarrow sent
\end{aligned}
$$

En la producción de S, id representa el identificador de la variable del ciclo. El primer *exp* es la expresión que nos da el valor inicial de la variable y el segundo representa el valor final que puede tomar. El símbolo no terminal I expresa incremento o decremento de la variable del ciclo de manera opcional.

(a) Obtener la gramática con atributos para realizar la comprobación de tipos.

(b) Obtener la gramática con atributos para generar código intermedio en cuartetos.

---

8. En lenguaje C la proposición `for` tiene la siguiente forma: `for (e1; e2; e3) S;`, entendiendo el siguiente significado:

```
e1 ;
while ( e2 )
{
    S ;
    e3 ;
}
```

Construir una definición dirigida por sintaxis para traducir dicha proposición a cuartetos.

---

9. Dada la siguiente gramática:

$$
\begin{aligned}
S &\rightarrow id\ (\ Lista\ ) \\
Lista &\rightarrow Expresion\ Lista \mid \varepsilon \\
Expresion &\rightarrow id \mid num \mid num.num \mid expresion\_compleja
\end{aligned}
$$

que representa la llamada a un procedimiento, obtener:

(a) La gramática con atributos para realizar la comprobación de tipos.

(b) La gramática con atributos para realizar su traducción dirigida por sintaxis a cuartetos.

---

10. Cuando los proposiciones generadas por la siguiente gramática se traducen a código en cuartetos la proposición `break` se traduce en un salto a la instrucción que va después de la proposición `while` englobadora más cercana. Para simplificar, las expresiones se representan mediante el símbolo terminal *exp* y los otros tipos de proposiciones con el terminal *otras*. Dichos terminales tienen un atributo sintetizado denominado `codigo` de tipo cadena de caracteres que almacena la traducción.

$$S \rightarrow \texttt{while}\ exp\ \texttt{do begin}\ S\ \texttt{end} \mid S\ ;\ S \mid \texttt{break} \mid otras$$

Obtener la traducción dirigida por sintaxis para cuartetos.

---

11. Sea la gramática que representa el acceso a un elemento de una matriz:

$$\begin{aligned} L &\rightarrow \text{id} \mid \text{id}[E] \\ E &\rightarrow E \mid E\ ,\ R \\ R &\rightarrow \text{id} \mid \text{exp} \end{aligned}$$

Obtener la traducción dirigida por sintaxis en cuartetos, teniendo en cuenta que el lenguaje en cuartetos sólo admite sentencias de la forma $id_1 := id_2[i]$ o $id_2 := id_1[i]$, donde id_2 es una matriz y representa la posición donde comienza la matriz e i es la posición relativa del elemento a partir de id_2. Suponer que las matrices se almacenan por columnas. El símbolo no terminal *exp* representa una expresión compleja, pero para simplificar se considera como un símbolo terminal que tiene asignados dos atributos `codigo` y `nombre` ambos de tipo cadena de caracteres donde el primero representa el código de la traducción y el segundo el nombre asignado.

---

12. Dada la gramática que genera expresiones lógicas:

$$\begin{aligned} E \rightarrow\ & E \text{ or } E \\ \mid\ & E \text{ and } E \\ \mid\ & \text{not } E \\ \mid\ & (E) \\ \mid\ & \text{id op\_relacional id} \\ \mid\ & \text{true} \\ \mid\ & \text{false} \end{aligned}$$

Escribir una traducción dirigida por sintaxis que obtenga el código en cuartetos de forma que, para generar las etiquetas, el traductor necesite una sola pasada. Para ello definir para E dos atributos sintetizados `listaverdad` y `listafalso` que contendrán las listas de etiquetas que quedan incompletas hasta el momento en el que se conoce su valor.

---

13. Ampliar la gramática del ejercicio anterior para que incluya las proposiciones de control de la siguiente forma:

$$\begin{aligned} S \rightarrow\ & \text{if } E \text{ then } S \\ \mid\ & \text{if } E \text{ then } S \text{ else } S \\ \mid\ & \text{while } E \text{ do } S \\ \mid\ & \text{begin } L \text{ end} \\ \mid\ & \text{otras} \\ L \rightarrow\ & L \ ;\ S \\ \mid\ & S \end{aligned}$$

donde ”otras” es un símbolo terminal que recoge las sentencias de asignación y lectura/escritura. Generalizar la traducción dirigida por sintaxis obtenida en el ejercicio anterior teniendo en cuenta la generación de etiquetas.

---

14. Se dice que una expresión está muy ocupada en un punto p del grafo de flujo si, con independencia del camino que se tome desde p, la expresión es evaluada antes de que se definan cualquiera de sus operandos.
    (a) Proporcionar un algoritmo de flujo de datos del estilo a los estudiados que permita encontrar las expresiones que se encuentran en esta situación.
    (b) ¿Qué operador de confluencia se utiliza?
    (c) ¿La propagación se realiza hacia adelante o hacia atrás?
