Apellido: ................................ Nombre: ......................................... Legajo:.............................

1er Parcial – **SINTAXIS Y SEMÁNTICA DE LENGUAJES**

13/06/2017 – Curso Martes - Turno Tarde – K2031

TEMA: 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1a | 1b | 1c | 2a | 2b | 2c | 3a | 3b | 4a | 4b | 4c | 4d | Notal Final |
| 1p | 1p | 1p | 1p | 1p | 1p | 1p | 1p | 0.5p | 0.5p | 0.5p | 0.5p |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

TIEMPO DISPONIBLE: 2 HORAS

**Ejercicio 1 – Gramáticas**

Dada la siguiente GIC (ver anexo):

1) Indique el orden de prioridad de los operadores y la asociatividad de los mismos justificando adecuadamente su respuesta.

De Mayor a Menor priodad: () , + , \*

A mayor distancia del axioma mayor prioridad

Asociativdad de los operadores:

+: Asociatividad derecha

\* : Asociatividad izquierda

La asociativdad se puede observar en el tipo de recursivdad que tiene la GIC.

2) Construya la tabla de derivación para la expresión 1+2\*(3+4+5) indicando la producción aplicada en cada paso.

Ver ejercicio página 38 , 39 , 40

3) Construya la tabla de evaluación (reducción) para la expresión 3+4+5 indicando la producción aplicada en cada caso.

Ver ejercicio página 38 , 39 , 40

**Ejercicio 2 – BNF**

Dada la siguiente BNF de sentencias y expresiones para ANSI C (ver anexo) indique si los siguiente constructos son derivables de la misma e indique si presentan errores sintácticos considerando las restricciones propias del lenguaje ANSI C. En caso de que sea derivable o de que no cumple alguna restricción justifique adecuadamente.

a) { { do {a;} while (a[(3)]) }

No es derivable de la BNF dada

* Falta una llave de cierre
* Falta ; luego de la sentencia do … while … ;

No presenta errores de restricciones de ANSI C. El valor del vector es convertido a booleano.

b) for (3;2==2;1) {a=-b; b=(3); 3=a;}

No es derivable de la BNF dada

* No se puede derivar ==

No cumple la restricción de la asignación. No es posible asignar un valor a un objeto que no es LValue, 3 es una constante.

c) while (a=3 && b=5) {;&a;3;for (;;) a}

No es derivable de la BNF dada

* No es posible derivar &&
* Falta un ; luego de la expresión “a” que se encuentra al final

No presenta errores de restricciones de ANSI C. Las asignaciones dentro del while son válidas y se convierten a valores booleanos para resolver la operación AND.

**Ejercicio 3 – Expresiones Regulares**

a) Escriba una ERX (metaER) que sea equivalente a la ER (a+b)\*(aa+bb)+a\*+b3

Una posible ERX: ([ab]\*((aa)|(bb)))|a\*|b{3}

b) Dada la siguiente ERX (metaER) (\+|\-)?[0-9]+ describa el lenguaje que representa.

Representa el lenguaje de los números enteros con o sin signo dado que no es opcional.

**Ejercicio 4 – Indique V o F**

a) La sintaxis de un lenguaje regular puede representare mediante una BNF

Falso. Sintaxis = BNF + RESTRICCIONES

b) En el lenguaje ANSI C, los lexemas main y while pertenecen a la misma categoría léxica

Falso. No pertenecen a la misma categoría, while es una palabra reservada, main es un identificador

c) En el lenguaje ANSI C, 0X451 y 0451 son dos constantes enteras distintas

Verdadero. Son constante enteras distintas, representan diferentes valores, una está en hexadecimal y otra en octal.

d) La metaER [ab]c? es equivalente a la metaER (bc)|b|a|(ac)

Verdadero. Representan el mismo lenguaje.

***Ejercicio 1***

1 Expresión -> Término |

2 Expresión \* Término

3 Término -> Factor |

4 Factor + Termino

5 Factor -> Número |

6 ( Expresión )

7 Número -> 1 | 2 | 3 | 4 | 5

***LEjercicio 2***

*sentencia:* una de *sentCompuesta sentExpresión sentSelección sentIteración sentSalto*

*sentExpresion: expresión* ***;***

*sentCompuesta:* **{** *listaDeclaraciones*op *listaSentencias*op **}**

*listaDeclaraciones: declaración*

*listaDeclaraciones declaración*

*listaSentencias: sentencia*

*listaSentencias sentencia*

*sentIteración:* **while (** *expresión* **)** *sentencia*

**do** *sentencia* **while (** *expresión* **) ;**

**for (** *expresión*op **;** *expresión*op **;** *expresión*op **)** *sentencia*

*expresión: expAsignación*

*expAsignación: expUnaria*

*expUnaria operAsignación expAsignación*

*operAsignación:* uno de **= +=**

*expUnaria: expPostfijo*

**++** *expUnaria*

*operUnario expUnaria*

**sizeof (***nombreTipo***)**

o*perUnario:* uno de **&** (dirección) **\*** (puntero) **–** (signo) **!** (“not”)

*expPostfijo: expPrimaria*

*expPostfijo* **[** *expresión* **]**

*expPostijo* **(** *listaArgumentos*op **)**

*listaArgumentos: expAsignación*

*listaArgumentos* **,** *expAsignación*

*expPrimaria: identificador*

*constante*

*literalCadena*

**(** *expresión* **)**

*nombreTipo:* uno de **char int double**