Departamento de Tecnologías de la Información

Área de Ciencias de la Computación e Inteligencia Artificial

Procesadores de lenguajes

Examen de junio

EJERCICIO 1 (2 puntos)

Considere la gestión de la memoria en tiempo de ejecución.

(a) ¿En que consiste la memoria de pila?

(b) Describa la estructura del registro de activación de una función.

(c) Describa el proceso de llamada a una función.

(d) Describa el proceso de retorno de una función.

EJERCICIO 2 (2 puntos)

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

c (**o** | b (**o** | c | b))* **c**

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

EJERCICIO 3 (2 puntos)

Considere la siguiente gramática, que describe las expresiones lógicas basadas en los operadores **and**, **or** y **not**:

Expr → *Expr* **or** *Term*

Expr → *Term*

Term → *Term* **and** *Comp*

Term → *Comp*

Comp → **not** *Base*

Comp → *Base*

Base → **id**

Base → **(** *Expr* **)**

Construya el autómata reconocedor de prefijos viables y la tabla de análisis SLR de la gramática planteada. Utilice para ello la tabla incluida en la última página.

EJERCICIO 4 (2 puntos)

La siguiente gramática describe expresiones formadas con las operaciones producto y potencia de números:

Expr → *Factor Producto*

Producto → **prod** *Factor Producto*

Producto → **lambda**

Factor → *Base Potencia*

Potencia → **power** *Base Potencia*

Potencia → **lambda**

Base → **num**

Base → **lparen** *Expr* **rparen**

Considere las siguientes clases que permiten definir expresiones con estos operadores:

```
// Clase abstracta que describe una expresión aritmética
public abstract class  Expression {
}

// Clase que describe un número constante
public class Number extends Expression {
  private double value;

  public Number(String val) {  this.value = Double.parseDouble(val); }
}

// Clase que describe la potencia entre dos expresiones
public class Power extends Expression {
  public Expression base;
  public Expression pow;

  public Power(Expression a, Expression b) {  this.base = a;   this.pow = b;  }
}

// Clase que describe el producto entre dos expresiones
public class Product extends Expression {
  public Expression left;
  public Expression right;

  public Product(Expression a, Expression b) {  this.left = a; this.right = b;  }
}
```

Desarrolle un ETDS que genere el árbol de sintaxis abstracta asociado a las expresiones formadas por productos y potencias de números. Es importante tener en cuenta que la potencia es un operador asociativo a la derecha, mientras que el producto es un operador asociativo a la izquierda.

EJERCICIO 5 (2 puntos)

La siguiente figura muestra la descripción sintáctica del bucle FOR basado en una progresión en el formato de la herramienta JavaCC (esta sintaxis se utiliza en el lenguaje Kotlin).

```
void InstFor() :
{}
{
  <FOR>
  <LPAREN>
    <ID> <IN> Range()
  <RPAREN>
  Instruccion()
}

void Range() :
{}
{
  <INTEGER_LITERAL>
  <DOTTED>
  <INTEGER_LITERAL>
  ( <STEP> <INTEGER_LITERAL> )?
}
```

Se pretende enriquecer la gramática anterior para construir el código intermedio de 3 direcciones correspondiente a la instrucción FOR basada en una progresión. Para ello la función *InstFor()* deberá devolver un objeto *Inst* cuyo campo *code* contenga dicho código. Por su parte, la función *Range()* deberá devolver la información de la progresión. Para representar esta información deberá crearse una clase auxiliar.

Considere que la función *Instruccion()* devuelve un objeto de la clase *Inst* cuyo campo *code* contiene el código intermedio asociado a a dicha instrucción. Se dispone del método *getNewLabel()*, que devuelve una nueva etiqueta (es decir, una etiqueta no utilizada en ningún punto del código) y del método *getNewTemp()* que devuelve una referencia a una nueva variable temporal.

Nota: Ejemplo de instrucción FOR utilizado en Kotlin.

```
for (x in 1..5 step 2) {
    print(x)
}
```

Estado	or	and	not	id	(	)	$	Expr	Term	Comp	Base