**Compiladores**

**Practica 1 Yacc Básico**

Escribir el código de soporte en lenguaje C o Java.

**Calculadora** para expresiones escritas en **postfijo**

Escribir una especificación de yacc para evaluar **expresiones aritméticas** escritas en notación **postfija** y que involucren números enteros

**Calculadora** para expresiones **booleana**s

Escribir una especificación de yacc para evaluar **expresiones booleanas** , que utilizan los operadores **AND** , **OR** y **NOT** , entre paréntesis ( ) , y los operandos verdadero, falso. Para el operador: **AND** (usar ^), **OR** (usar v), y **NOT** (usar -). Para los operandos: verdadero (usar **t**) y falso (usar **f**).

listados de menor a mayor precedencia **OR**, **AND**, **NOT**

Suponga que cuenta con el código de las operaciones para : números **racionales**, números **complejos** y **polinomios**.

Escriba una **calculadora** para : números **racionales,** onúmeros **complejos,** o **polinomios**

Para esto escriba una especificación de yacc para evaluar expresiones que involucren operaciones con : números **racionales** o números **complejos** o **polinomios**

**Calculadora** para **conjuntos**

Suponga que cuenta con el código de la unión, la intersección , la diferencia , el complemento , potencia. Escribir una especificación de yacc para evaluar expresiones que involucren operaciones con **conjuntos**.

**Calculadora** para **vectores**

Suponga que cuenta con el código de el producto punto, el producto punto cruz , la multiplicación por un escalar, la suma, la resta y la magnitud. Escribir una especificación de yacc para evaluar expresiones que involucren operaciones con **vectores**.

*Entrada*

[ 1 2 3 ] + [ 2 4 6 ]

*Salida*

[ 3 6 9 ]

*Entrada*

[ 1 2 3 4 5 ] + [ 2 4 6 8 10]

*Salida*

[ 3 6 9 12 15 ]

**Calculadora** para **matrices**

Suponga que cuenta con el código de la multiplicación , la suma, la resta , la multiplicación por un escalar, y la inversa. Escribir una especificación de yacc para evaluar expresiones que involucren operaciones con **matrices**.

**Robot**

Escribir una especificación de yacc para calcular e imprimir la posición final del robot

|  |  |  |
| --- | --- | --- |
| Producción | Regla Semántica | |
| *sec ->* **comienza** | *sec.x*  *sec. y* | = 0 = 0 |
| *sec -*> *sec instr* | *sec.x sec. y* | = *sec.x + instr.dx = sec.y + instr.dy* |
| *instr -*> **este** | *instr.dx instr.dy* | = 1  = 0 |
| *instr -*> **norte** | *instr.dx instr. dy* | = 0  = 1 |
| *instr -*> **oeste** | *instr.dx i instr.dy* | = -1 = 0 |
| Instr -> **sur** | *instr.dx instr.dy* | = 0  = -1 |

**Tabulador de funciones polinomiales**

Escribir una especificación de yacc para tabular funciones polinomiales construya el **árbol sintáctico** que representa la expresión y luego evalúelo para cada valor de la variable x.

Nota: Utilice un rango por default donde no se indetermine la función

**Tablas de verdad**

Escribir una especificación de yacc para imprimir la **tablas de verdad** de **expresiones booleanas** de n variables. Nota la tabla tendrá 2n renglones.

**Traductor español-ingles**

Debe reconocer una parte muy pequeña de la gramática del lenguaje español y tiene que contar con un diccionario español-ingles y debe clasificar las palabras en **ARTICULOS**, **SUSTANTIVOS**, **ADJETIVOS** **VERBOS** y **ADVERBIOS**.

ejemplo

el perro grande come

the big dog eats

**Conversador**

Debe reconocer una parte muy pequeña de la gramática del lenguaje español y tiene que contar con un diccionario de español y debe clasificar las palabras en **SUSTANTIVOS**, **VERBOS**, **DETERMINANTES**, **DONDE**, **PREPOSICIONES**, y **ADJETIVOS**.

ejemplos

oración sujeto predicado

quien verbo

Eva corre en el parque

donde corre Eva?

en el parque

Juan come en la fonda

donde come Juan ?

en la fonda

Pedro estudia

quien estudia?

Pedro estudia

**Derivador Simbólico**

Obtener la derivada de expresiones formadas mediante los operadores aritméticos + y \* , la variable x, y números enteros. se puede ir construyendo el árbol de la expresión y el árbol de su derivada al mismo tiempo. Nota no es necesario efectuar ninguna simplificación.

variable x, polinomios , arboles

**generador de código en ensamblador**

Escribir una especificación de yacc para generar un programa (extension .com) en ensamblador para evaluar una **expresión aritmética** escrita en **postfijo**.

**generador de GUI**

Escribir una especificación de yacc para generar un programa en **Java** que tenga la **GUI** especificada

Se consideran solo botones y etiquetas

con las siguientes propiedades

nombre (lo que ve el programador)

texto (lo que ve el usuario)

posición x, y

color background, foreground

alto, ancho

Lenguaje **Litle quilt (colchita)**

El lenguaje se define por la siguiente gramática BNF:

<QUILT> ::= **A** | **B** | **turn(**<QUILT>**)** | **sew(**<QUILT>,<QUILT>**)**

**A** y **B** representan los dos colchas primitivas. Cada colcha primitiva corresponde a un arreglo matricial de 2 × 2 caracteres. **girar ()** y **coser ()** son las operaciones aplicables a los edredones.

La instrucción **girar(**x**)** gira la colcha x 90 grados hacia la derecha. La siguiente tabla ilustra los edredones primitivos, así como ejemplos de los efectos de la operación de **girar()**:

En consecuencia, la instrucción **coser (**x, y**)** cose la colcha x a la izquierda de la colcha y. Ambos x e y deben tener la misma altura, de lo contrario se generará un error. La siguiente figura representa el resultado de **coser (A, girar(B))**:

// ||

/+ ||

mientras que **coser(girar (coser (B, girar (B))), A)** genera un mensaje de error.

Su trabajo es construir un intérprete del lenguaje de **Little Quilt**.

**Consultas sobre perros**

Consultar datos de un perro (almacenados en una lista simplemente ligada) por nombre , raza, edad , o genero

Obtener los datos de todos los perros de nombre fido

Obtener los datos de todos los cachorros

Obtener los datos de todos los dalmata