Grupo Miércoles 12:00-14:00 semanas A
Práctica 4
Autor: Diego Marco

newick.l :
He creado las reglas sintacticas “(“ , “)”, ”,” , ”:” y “;” para reconocer los separadores del formato Newick. La acción es devolver los token OP, CP, COMA, COLON, SEMICOLON, respectivamente.
He creado la regla [a-z|A-Z]+[a-z|A-Z|_|0-9]* para reconocer el nombre de los nodos. La acción relacionada con esta regla es guardar su valor semántico (char*) y devolver el token NODE.
He creado la regla [0-9]+\.?[0-9]* para reconocer las distancias de los nodos, que pueden estar en notacion “1.0” o “1”. La acción relacionada con esta regla es guardar su valor semántico (double) y devuelve el token HIGH.

newick.y:
He declarado el tipo “datos” que es un struct que contiene un dato de tipo “double” y un dato de tipo “char*”, usando %type he declarado de este tipo los símbolos no terminales” árbol, contenido_1, contenido_2, contenido_3, existe_nodo, existe_distancia “. Y los símbolos terminales “HIGH, NODE”.
He creado la gramática que genera árboles en formato newick.

Pruebas:
Entrada:
((a,a)j:8,((a,b),(:89,b,c))a,((a,b),(,b)),b)tyr;

Salida:
La altura es 92.00
La raiz se llama tyr
El numero de hojas es 12

Entrada:
(a,a);

Salida:
La altura es 2.00
La raiz se llama
El numero de hojas es 2

Entrada:
(,,(,));

Salida:
La altura es 3.00
La raiz se llama
El numero de hojas es 4

Entrada:
((B:0.2,(C:0.3,D:0.4)E:0.5)A:0.1)F;

Salida:
La altura es 2.00
La raiz se llama F
El numero de hojas es 3