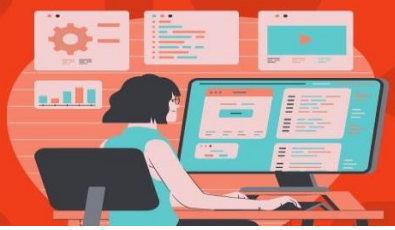


LENGUAJES Y AUTÓMATAS II

ME XENIA PADILLA MADRID

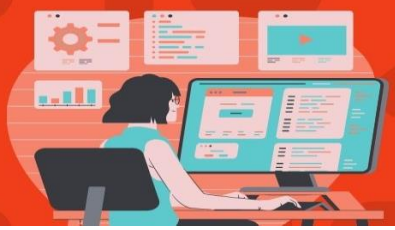
ACTIVIDAD DE LABORATORIO	**03**
NOMBRE DE LA ACTIVIDAD	Mi Lenguaje
OBJETIVO DE APRENDIZAJE	Crear una gramática libre de contexto con la capacidad de reconocer un lenguaje de programación básico.
INSTRUCCIONES	Utiliza ANTLR para generar un reconocedor del Lenguaje C
NOMBRE ESTUDIANTE	Itsai Zempoaltecatl Mejia
GRUPO	6SA
FECHA	17/febrero/2022

Lenguaje.g4

```
grammar Lenguaje;

@parser::header {
    import java.util.HashMap;
}

@parser::members {
   HashMap<String,Integer> memoria = new H
ashMap<String,Integer>();
}

programa: VOID MAIN '(' ')' '{' body '}';

body: declaracion* | asignacion*;

declaracion: 'int' ID ';';

asignacion: ID '=';

VOID: 'void';
MAIN: 'main';
ID: [a-zA-Z]+;
WS: [ \t\r\n]+ -> skip;
```

1. Utiliza la gramática Lenguaje.g4 (o la versión que uses)
2. Agrega la producción **asignación**
3. La sintaxis para la producción de asignación es: **asignación: ID = expresión**
4. Agrega la gramática de **expresión**

PRODUCTO A ENTREGAR: Colocar dentro de la forma el léxico elaborado y la salida obtenida con una descripción de lo elaborado. **Todo dentro de la forma**



LENGUAJES Y AUTÓMATAS II

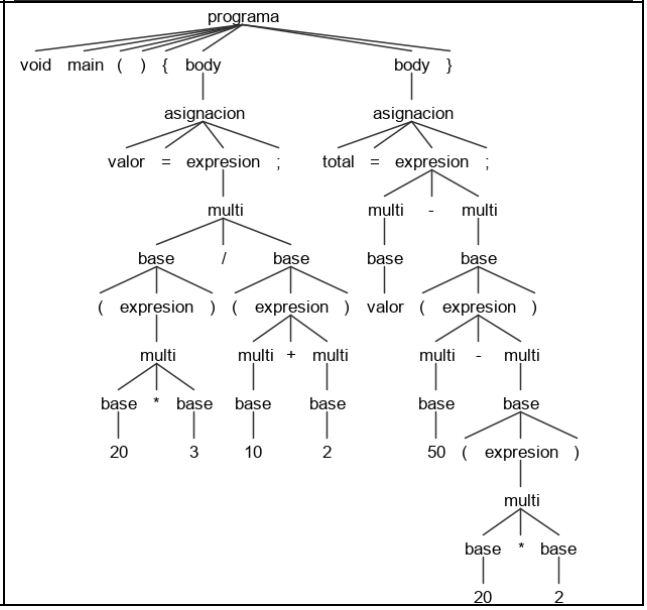
ME XENIA PADILLA MADRID

TABLA DE RESULTADOS

GRAMÁTICA (Solo la sección nueva generada)	Árbol sintáctico con la entrada

```
void main () {
valor = (20*3) / (10+2);
total = valor - (50 - (20*2));
}
```

```
asignacion  : ID '=' expresion+ SEMI;
expresion   : multi (('+' | '-') multi)*;
multi       : base (('*' | '/') base)*;
base        : ID | NUM | '(' expresion ')';
NUM: [0-9]+;
```



CONCLUSIÓN: Este ejercicio usa algo de recursividad, ya que base manda a llamar a expresión aunque expresión tenga mas jerarquía que base, pero al evaluar el texto después del '=' y encontrar paréntesis es necesario evaluar de arriba para abajo