1) Interpretación. El lenguaje L (3 puntos)

L es un lenguaje de programación secuencial con tipado dinámico de sus variables (las variables toman el tipo de las expresiones asignadas). El programa L está compuesto por una secuencia de instrucciones (asignación y muestra de valores de variables por consola). Las expresiones en L son de dos tipos: (a) enteras y (b) listas. Los elementos de una lista pueden ser enteros u otras listas. Las expresiones enteras, además de las operaciones convencionales, incluyen la selección de elementos enteros de una lista. Las expresiones listas incluyen listas por extensión, segmentos de listas definidos con rangos de índices y selección de elementos listas de una lista. Para precisar y completar la definición sintáctica del lenguaje, se adjunta su gramática a continuación:

```
programa: PROGRAMA instrucciones FPROGRAMA

instrucciones: instruccion instrucciones
   | instruccion

instruccion: asignacion | mostrar

asignacion: IDENT ASIGNACION expresion PUNTOYCOMA

mostrar: MOSTRAR PARENTESISABIERTO variables PARENTESISCERRADO PUNTOYCOMA;

variables: variable COMA variables
     | variable

expresiones: expresion COMA expresiones
       | expresion

expresion: expresion_entera
   | expresion_lista
   ;

expresion_entera: expresion_entera MAS expresion_entera
   | expresion_entera MENOS expresion_entera
   | expresion_entera POR expresion_entera
   | expresion_entera DIVISION expresion_entera
   | PARENTESISABIERTO expresion_entera PARENTESISCERRADO
   | NUMERO
   | variable

expresion_lista: CORCHETEABIERTO (expresiones)? CORCHETECERRADO
   | PARENTESISABIERTO expresion_lista PARENTESISCERRADO
   | segmento_lista
   | variable

segmento_lista: IDENT CORCHETEABIERTO rango CORCHETECERRADO

rango: expresion_entera RANGO expresion_entera

variable : IDENT CORCHETEABIERTO expresion_entera CORCHETECERRADO
   | IDENT
```

A continuación, se muestra un programa L de ejemplo (con anotaciones sobre su interpretación):

```
PROGRAMA
  x = 1;                  // x toma el valor 1
  mostrar(x);             // mostrar por consola 1
  x = [3,x,12,[1,x+5]];   // x toma el valor [3,1,12,[1,6]]
  y = x[1]+2;             // y toma el valor 5
  mostrar(x,y);           // mostrar por consola [3,1,12,[1,6]], 5
  y = x[x[1]-2..x[1]];    // y toma el valor [3,1]
  mostrar(y);             // mostrar por consola [3,1]
FPROGRAMA
```

SE PIDE:

Decisiones y gramática atribuida de un intérprete de L (2 puntos).

```
OBJETIVO
--------
   Construir un intérprete del lenguaje L.

DECISIONES
----------
   1) Memoria para almacenar variables con sus valores
      Ejemplo:
            variable  |     valor
            ----------------------------
               x      | [3,1,12,[1,6]]
               y      |     [3,1]
               z      |       3

   2) Actualizar la memoria de variables con el valor de las expresiones
      en las asignaciones.

   3) Calcular (recursivamente) el valor de una expresión.

      expresion dev valor: valor=expresion_entera
         | valor=expresion_lista

      expresion_entera dev valor: valor1=expresion_entera MAS valor2=expresion_entera
           {valor = sumar valor1 y valor2}
         | valor1=expresion_entera MENOS valor2=expresion_entera
           {valor = restar valor2 de valor1}
         | valor1=expresion_entera POR valor2=expresion_entera
           {valor = multiplicar valor1 y valor2}
         | valor1=expresion_entera DIVISION valor2=expresion_entera
           {valor = dividir valor1 y valor2}
         | PARENTESISABIERTO valor=expresion_entera PARENTESISCERRADO
         | NUMERO {valor=el valor de NUMERO}
         | valor=variable

      expresion_lista dev valor:
           CORCHETEABIERTO (seq_valores=expresiones)? CORCHETECERRADO
           { valor = lista con seq_valores como elementos }
         | PARENTESISABIERTO valor=expresion_lista PARENTESISCERRADO
         | valor=segmento_lista
         | valor=variable
```

```
      segmento_lista dev valor:
         IDENT CORCHETEABIERTO (ext_inf,ext_sup)=rango CORCHETECERRADO
         {valor = lista construida con los elementos de IDENT en memoria
                  contenidos entre las posiciones ext_inf y ext_sup excluyendo
                  este último elemento}

      rango dev ext_inf, ext_sup:
         ext_inf=expresion_entera RANGO ext_sup=expresion_entera ;

      variable dev valor:
           IDENT CORCHETEABIERTO valor1=expresion_entera CORCHETECERRADO
           {valor=selecciona el elemento de IDENT en memoria correspondiente
                  a la posición valor1}
         | IDENT {valor=consultar el valor de IDENT en memoria}

GRAMATICA
---------
(global)
    memoria_variables

programa: PROGRAMA instrucciones FPROGRAMA

instrucciones: instruccion instrucciones
   | instruccion

instruccion: asignacion | mostrar

asignacion: IDENT ASIGNACION valor=expresion PUNTOYCOMA
   {actualizar la memoria de variables con valor para IDENT}

mostrar: MOSTRAR PARENTESISABIERTO seq_valores=variables PARENTESISCERRADO PUNTOYCOMA
   {mostrar por pantalla seq_valores}

variables dev seq_valores:
   valor=variable {añadir valor al final de seq_valores}
   COMA valores_aux=variables {añadir valores_aux al final de seq_valores}
 | valor=variable {añadir valor al final de seq_valores}

expresiones dev seq_valores:
         valor=expresion {añadir valor al final de seq_valores}
         COMA seq_aux=expresiones {añadir seq_aux al final de seq_valores}
       | valor=expresion {añadir valor al final de seq_valores}

expresion dev valor: valor=expresion_entera
         | valor=expresion_lista

  expresion_entera dev valor: valor1=expresion_entera MAS valor2=expresion_entera
       {valor = sumar valor1 y valor2}
     | valor1=expresion_entera MENOS valor2=expresion_entera
       {valor = restar valor2 de valor1}
     | valor1=expresion_entera POR valor2=expresion_entera
       {valor = multiplicar valor1 y valor2}
     | valor1=expresion_entera DIVISION valor2=expresion_entera
       {valor = dividir valor1 y valor2}
     | PARENTESISABIERTO valor=expresion_entera PARENTESISCERRADO
     | NUMERO {valor=el valor de NUMERO}
     | valor=variable
```

```
  expresion_lista dev valor:
       CORCHETEABIERTO (seq_valores=expresiones)? CORCHETECERRADO
       { valor = lista con seq_valores como elementos }
     | PARENTESISABIERTO valor=expresion_lista PARENTESISCERRADO
     | valor=segmento_lista
     | valor=variable

  segmento_lista dev valor:
     IDENT CORCHETEABIERTO (ext_inf,ext_sup)=rango CORCHETECERRADO
     {valor = lista construida con los elementos de IDENT en memoria
              contenidos entre las posiciones ext_inf y ext_sup excluyendo
              este último elemento}

  rango dev ext_inf, ext_sup:
     ext_inf=expresion_entera RANGO ext_sup=expresion_entera ;

  variable dev valor:
       IDENT CORCHETEABIERTO valor1=expresion_entera CORCHETECERRADO
       {valor=selecciona el elemento de IDENT en memoria correspondiente
              a la posición valor1}
     | IDENT {valor=consultar el valor de IDENT en memoria}
```

Implementación Antlr4 mediante métodos visitors de las reglas `expresion_lista` y `segmento_lista` de la gramática atribuida resultante (1 punto).

```
/*
expresion_lista dev valor:
   CORCHETEABIERTO (seq_valores=expresiones)? CORCHETECERRADO
   { valor = lista con seq_valores como elementos }
 | PARENTESISABIERTO valor=expresion_lista PARENTESISCERRADO
 | valor=segmento_lista
 | valor=variable
 */
@Override
public Object visitExpresion_lista(Anasint.Expresion_listaContext ctx) {
    Object valor=null;
    switch(ctx.getChildCount()){
        case 3:
            if (ctx.getChild(0).getText().equals("[")){
                List<Object>seq_valores=new LinkedList<>();
                seq_valores.addAll((List<Object>)visitExpresiones(ctx.expresiones()));
                valor=new LinkedList<>();
                ((List<Object>)valor).addAll(seq_valores);
            }
            else
                valor=visitExpresion_lista(ctx.expresion_lista());
            break;
        case 2:
            valor=new LinkedList<>();
            break;
        default:
            if (ctx.segmento_lista()!=null)
                valor=visitSegmento_lista(ctx.segmento_lista());
            else
                valor=visitVariable(ctx.variable());
    }
    return valor;
}
```

```
/*
segmento_lista dev valor:
 IDENT CORCHETEABIERTO (ext_inf,ext_sup)=rango CORCHETECERRADO
 {valor = lista construida con los elementos de IDENT en memoria
          contenidos entre las posiciones ext_inf y ext_sup excluyendo
          este último elemento}
 */
@Override
public Object visitSegmento_lista(Anasint.Segmento_listaContext ctx) {
    Object valor=new LinkedList<>();
    Pair<Integer,Integer> tupla = (Pair<Integer,Integer>)visitRango(ctx.rango());
    Integer ext_inf=tupla.a-1;
    Integer ext_sup=tupla.b-1;
    List<Object>aux=(List<Object>)memoria_variables.get(ctx.IDENT().getText());
    aux=aux.subList(ext_inf,ext_sup);
    ((List<Object>)valor).addAll(aux);
    return valor;
}
```

NOTA: sea claro y legible.