Práctica. Tercera Fase

Desarrollo de constructores de ASTs para Tiny

En esta tercera parte debe realizarse el desarrollo de constructores de ASTs descendentes y ascendentes para **Tiny**, así como implementar un procesamiento sencillo. Para ello, deberá entregarse:

- Una memoria con las siguientes secciones:
  - Portada en la que aparezcan los nombres y apellidos de los integrantes del grupo, y el número de grupo.
  - Especificación de la sintaxis abstracta de **Tiny** mediante la enumeración de las signaturas (cabeceras) de las funciones constructoras de ASTs.
  - Especificación del constructor de ASTs mediante una gramática s-atribuida.
  - Acondicionamiento de dicha especificación para permitir la implementación descendente.
  - Especificación de un procesamiento que imprima los tokens del programa leído, uno en cada línea, pero omitiendo los paréntesis redundantes en las expresiones (es decir, aquellos paréntesis que, eliminados, no cambian el significado de dichas expresiones). Las palabras reservadas se escribirán con todas las letras en minúscula, y entre ángulos (<…>). El fin de fichero se escribirá como <EOF>.
- Una implementación orientada a objetos en Java de la sintaxis abstracta de Tiny, preparada tanto para soportar **programación recursiva**, como para soportar procesamiento mediante el patrón **visitante**.
- Una implementación descendente del constructor de ASTs desarrollada con **javacc**.
- Una implementación ascendente de dicho constructor desarrollada con CUP y **jflex**.
- Tres implementaciones del procesamiento pedido: (i) una utilizando **programación recursiva**; (ii) una utilizando el patrón **intérprete**; (iii) una tercera utilizando el patrón **visitante**.
- Un programa principal que integre ambos constructores, y también los procesamientos desarrollados. Dicho programa recibirá como argumentos (i) el archivo a analizar; (ii) una opción *opc* que indique el constructor de ASTs a aplicar (si *op* es `desc` el constructor a aplicar será el descendente; si es `asc` será el ascendente); y (iii) una opción *opp* que indica que estilo de procesamiento a aplicar (si *op* es `rec` se aplicará programación recursiva, si *op* es `int` se aplicará el procesamiento basado en el patrón intérprete, si es `vis` el basado en el patrón `visitante`). El programa producirá como salida, bien un mensaje legible del primer error (léxico o sintáctico) detectado, bien una impresión, en los términos indicados, del programa leído en caso de que no se hayan detectado errores.

Fecha límite de entrega: **Viernes 12 de abril de 2021, a las 11:59 pm**.

Modo de entrega: A través del campus virtual, en un único .zip. Dicho archivo debe contener: (i) un documento PDF `memoria.pdf` con la memoria requerida; (ii) una carpeta `implementación`, en el interior de la cuál debe incluirse toda la implementación requerida; y (iii) una carpeta `pruebas` con distintos programas de prueba que permitan probar la implementación. La entrega debe ser realizada solamente por un miembro del grupo.

Las implementaciones deberán, además, entregarse a través del juez DomJudge de la asignatura, siguiendo las indicaciones que se proporcionarán más adelante.