UTN FRBA – SSL – Examen Final – 2018-12-03

Apellido, Nombre:		Legajo:		Nota:	

- Resuelva el examen en tinta y en esta hoja; no se aceptan hojas adicionales.
- Para los ítems de *una mejor respuesta*, marcados con una círculo (❍), tilde (✓) sólo una opción, la mejor.
- Para los ítems de *respuestas múltiple*, marcados con un caja (❑), tilde (✓) todas las respuestas correctas.
- Durante el examen no se responde consultas; si lo necesita, escriba hipótesis de trabajo, las cuales también se evalúan.

1. *(2 puntos)* Tilde todas las afirmaciones **verdaderas** con respecto a los LR:
   - ❑ Son fundamentales para los LP.
   - ❑ Son representables mediante ER.
   - ❑ Son representables mediante BNF.
   - ❑ Son cerrados bajo la concatenación.
   - ❑ Pueden incluir sublenguajes que no son LR.
2. *(2 puntos)* Tilde todas las afirmaciones **verdaderas** con respecto a las GR:
   - ❑ Pueden generar lenguajes infinitos.
   - ❑ Generan lenguajes representables por RegEx.
   - ❑ Sus producciones pueden tener ε en su derecha.
   - ❑ Generan LF reconocibles con autómatas con pila.
   - ❑ Sus producciones pueden tener terminales en su izquierda.
3. *(2 puntos)* Analice la siguiente afirmación sobre las GIC LL(1): Dadas las producciones de un no terminal, los conjuntos primeros de los lados derechos deben ser disjuntos.
   ¿Está de acuerdo con la afirmación? ¿Por qué?:

4. Sea `int i=1;void*p=&i;` describa el **error semántico** en cada sentencia o escriba *correcto* si no lo hay:
   a. *(1 punto)* `{ ++p; }`
   b. *(1 punto)* `{ *p=i; }`
   c. *(1 punto)* `{ double i=42; i=p*i; }`
   d. *(1 punto)* `{ int *p=malloc(sizeof i); }`
   e. *(Punto extra)* Describa el posible error pragmático de una de las sentencias.

1. Una Resolución

1.
   - ✓
   - ✓
   - ✓
   - ✓
   - ✓
2.
   - ✓
   - ✓
   - ✓
   - ✓
   - ❏
3. Sí, porque si no son disjuntos no hay forma de seleccionar la producción a aplicar con solamente un (1) siguiente token.
4.
   a. No es posible determinar el siguiente objeto.
   b. No es posible desreferenciar un puntero a `void`.
   c. El operador binario `*` requiere tipos aritméticos. Nota: el `double` simplemente oculta el `int`.
   d. Correcto. Nota: el `int*` simplemente oculta el `void*`.
   e. Luego de la sentencia compuesta, no hay forma de liberar el bloque reservado por `malloc`, hay memory leak.

v1.0.0-rc.1, 2018-12-02