UTN FRBA – SSL – Examen Final – 2020-03-03

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. *(1 punto)* Escriba una RegEx (metaER) que represente las constantes hexadecimales sin sufijo en C:
2. Dado el fragmento: `0xF+F0x+0L+L0`
   a. Analícelo **léxicamente**:
      i. *(1 punto)* Indique cuantos **tokens** tiene:
      ii. *(1 punto)* Indique cuantos **tipos de tokens** o categorías léxicas tiene. **Justifique**:
      iii. *(1 punto)* Indique cuantas invocaciones a `getchar` se necesitan para su análisis:
      iv. *(1 punto)* Indique cuantas invocaciones a `ungetc` se necesitan para su análisis:
   b. *(1 punto)* Indique la categoría **sintáctica** a la que pertenece el fragmento: Declaración, Expresión, Sentencia o Error sintáctico.
   c. *(2 puntos)* Escriba la o las declaraciones necesarias para que sea un fragmento **semánticamente correcto** con tipo `double` y valor 15.
   d. *(1 punto)* Escriba la o las declaraciones necesarias para que sea un fragmento **semánticamente incorrecto** por **error de tipo**.
3. *(1 punto)* Indique el valor de verdad de la siguiente arfimación y **justifique**: La sintaxis de C especifica el orden de evaluación de los operandos y la precedencia de los operadores.
4. *(Punto extra)* Escriba un fragmento breve que sea sintáctimente válido tanto en *C* y como en *C++*, pero que tenga diferente semántica para cada LP.

1. Una Resolución

1. `0[xX][0-9A-Fa-f]+`
2. a. i. 7
      ii. 3. Categorías: constante, identificador, adición.
      iii. 22
      iv. 7
   b. Expresión.
   c. `double F0x=0,L0=0;`
   d. `struct {double x,y;} F0x,L0;`, no es posible sumar tipos `struct`.
3. Falso. La semántica especifica el orden de evalaución de los operandos, las sintaxis especifica asociatividad de los operadores y precedencia de los operadores.
4. `struct Punto{double x,y;};`

v1.0.1-beta.1, 2020-03-04