

## UTN FRBA – SSL – Examen Final – 2023-02-27

Apellido, Nombre:		Legajo:		Nota:	
-------------------	--	---------	--	-------	--



- Resuelva el examen en el documento compartido para edición; no se aceptan documentos adicionales.
- Durante el examen no se responden consultas; si lo necesita, escriba hipótesis de trabajo, las cuales también se evalúan.

1. Dado el siguiente fragmento

```
if(a>0)
    if(a<1)
        b++;
else
    c++;
```

- (1 punto) ¿Cuántas llamadas a `getchar` requiere el scanner para generar el primer token? Justifique.
  - (2 puntos) Indique si es sintácticamente correcta, justifique con árboles de derivación.
  - (1 punto) Declare `c` para que la última línea sea semánticamente **incorrecta**.
2. Dado  $\Sigma = \{a, b\}$  y el LF las palabras que terminan con `a`:
- (1 punto) Escriba una regex que lo represente.
  - (1 punto) Escriba una regex que represente el complemento del LF.
  - (1 punto) Indique si el LF o su complemento son sublenguajes del LF *identificadores de C*.
  - (1 punto) Escriba el prototipo de una función que retorne si una cadena es palabra del LF, utilice `const` correctamente.
  - (2 puntos) Codifique en C una tabla de transición para un AFD que reconozca el LF.
3. (Punto extra) Declarar `Stack` para que la expresión `Stack.Push(3.14)` sea semánticamente correcta.

## 1. Una Resolución

1.

- a. Tres: i, f, y (. Después de la f debe seguir leyendo.
- b. La gramática es ambigua, por lo que hay dos árboles posibles según BNF, pero C resuelve la ambigüedad tomando el else como parte del if más léxicamente cercano y que genera sintácticamente sea correcto. Así que hay un solo árbol posible que se pueda armar con esta secuencia de tokens y si se puede derivar, es sintácticamente correcta.

c. `int c[7];`

2.

- a. `(a|b)*a`
- b. `((a|b)*b)?`
- c. El LF sí, porque la regex de identificadores representa todas las palabras del LF y más. El complemento no porque contiene la palabra vacía que no es un identificador válido.
- d. `bool EsPalabra(const char*);`
- e. Hay varias formas, esta es una:

<code>0-&gt;a-&gt;1</code>	<code>0-&gt;b-&gt;2</code>
<code>1-&gt;a-&gt;1</code>	<code>1-&gt;b-&gt;2</code>
<code>2-&gt;a-&gt;1</code>	<code>2-&gt;b-&gt;2</code>

El inicial es 0, y 1 es el único final

// se podría agregar una columna para otros símbolos, que lleve a un estado de rechazo

```
int t[][2]={
    {1,2},
    {1,2},
    {1,2}
};
```

3.

```
struct {
    void (*Push)(double);
} Stack;
```

v1.0.0 2023-04-08