

UTN FRBA – SSL – Examen Final – 2017-09-26

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



- Resuelva el examen en tinta y en esta hoja; no se aceptan hojas adicionales.
- Durante el examen no se responde consultas; si lo necesita, escriba hipótesis de trabajo, las cuales también se evalúan.
- Para los ítems de *selección múltiple*, tilde (✓) sólo una opción, la mejor.

1. (3 puntos) Sean los siguientes tres AFD:

$M_1 = (Q_1, \Sigma, T_1, q_1, F_1)$ y

$M_2 = (Q_2, \Sigma, T_2, q_2, F_2)$

$M = M_1 \cap M_2 = (Q, \Sigma, T, q_0, F)$

Defina $Q, q_0, F, T(q, x)$ de M :

$Q =$

$q_0 =$

$F =$

$T(q, x) =$

2. (2 puntos) Dada la siguiente GIC:

Expresión \rightarrow *Expresión* *Operador* *Número* | *Número*

Operador $\rightarrow + \mid - \mid * \mid /$

Donde *Expresión* es el axioma y *Número* representa las constantes enteras de \mathbb{C} , indique si el lenguaje generado por la GIC es regular, justifique:

3. (1 punto) En el contexto de la sintaxis de las expresiones, indique cuál concepto no es definido por el BNF de las expresiones:

- ☐ Precedencia de operadores.
- ☐ Token que representa cada operador.
- ☐ Orden de evaluación de cada operando.
- ☐ Cantidad de operandos de cada operador.
- ☐ Asociatividad de izquierda a derecha de cada operador.

4. (2 puntos) Indique el valor de verdad de la siguiente afirmación, y *justifique*:

El programa *yacc* genera un parser que invoca una vez a *yytlex* para obtener la secuencia de tokens del programa.

Justificación:

5. Dada la expresión $a[f()]$:

- (1 punto) Escriba **una** declaración que declare **ambos** identificadores, y que haga que la expresión sea semánticamente **correcta**.
- (1 punto) Escriba **una** declaración que declare **ambos** identificadores, pero que haga que la expresión sea semánticamente **incorrecta**.

1. Resolución

1.

$$Q = Q_1 \times Q_2$$

$$q_0 = (q_1, q_2)$$

$$F = F_1 \times F_2$$

$$T(q, x) = T((q_1, q_2), x) = (T_1(q_1, x), T_1(q_2, x))$$

2.

Número es regular porque pertenece al nivel léxico de C. *Operador* es finito, entonces es regular. *Expresión* produce clausuras y concatenaciones de lenguajes regulares, los LR son cerrados en esas operaciones. Por lo tanto la GIC genera un LR.

También es posible encontrar una GR, un AF, o una ER; lo dejamos como ejercicio.

3.

✓ Orden de evaluación de cada operando.

4. Falso: El parser invoca a `yyllex` cada vez que requiere un token.

5.

a. `int a[42], f(void);`

b. `int a, f;`