1 (2.5 ptos) Dada la siguiente gramática que permite generar declaraciones de variables simples y matrices de cualquier dimensión en el lenguaje C:

D → T L

T → **int** | **float**

L → L , I | I

I → I **[num]** | **id**

a) Construye la tabla de análisis sintáctico descendente. Transforma la gramática si es necesario para que sea LL(1). b). Indica el conjunto de predicción y de sincronización de cada símbolo no-terminal. c). Indica el contenido de la pila y de la entrada para la cadena `int a[3][10].`

2 (2.0 ptos) Considérese la siguiente gramática ambigua para *n* operadores infijos binarios:

E → E θ_1 E |…| E θ_n E | **(** E **)**| **id**

Supóngase que todos los operadores son asociativos por la izquierda y que θ_i tiene mayor prioridad que θ_j si $i > j$. Obtén una gramática no ambigua equivalente según las prioridades que se indican.

3 (2.5 ptos) Dada la siguiente gramática:

S → **if** C **then** S **else** S | **if** C **then** S | **other**

C → **0** | **1** | C **and** C

Se pide: a) ¿Es una gramática SLR(1)?. Construye la tabla de análisis sintáctico. En caso de que haya conflictos elige la opción adecuada para que el `else` esté asociado al `if` más cercano y el operador `and` sea asociativo a izquierdas. b) Haz la traza para reconocer la cadena `if 0 and then other`. Indica el mensaje de error y cómo se realiza la recuperación en modo de pánico.

4 (3.0 ptos) Dada la construcción de programación del bucle `for` de Matlab

S → **for id = num:num** S* **end**

Se pide: a) Dibuja el diagrama de flujo. b). Indica la forma del árbol que construirías. c) Implementa la función genera código. d). Construye el árbol de análisis sintáctico e indica la lista de cuádruplos para este ejemplo.

```
i = 0;
for k=1:10
     i = i + 1;
     print i;
end;
```