1. ¿Qué calcula el siguiente programa escrito en notación de Bison?

```
%{
#include ‘‘lex.yy.c’’
%}
%union {
int entero;
}
%token VAR
%token ID
%type<entero> declaration
%type<entero> variable_list
%start declaration
%%
declaration:  VAR variable_list { printf(‘‘%d ‘‘, $2); } ;
variable_list:  ID variable_list { $$ = 1+$2; } | { $$=0; } ;
%%
main()
{
yyparse();
}
```

   ¿ Qué imprime para la entrada: `VAR a b c` ? Amplia y modifica el programa añadiendo el código C para que imprima el lexema de cada variable declarada. No usar variables globales.

2. Dada la siguiente gramática escrita en Bison y suponiendo que tenemos las funciones `haznodo(...)` y `unir_nodos(...)` ya implementadas, introduce las acciones necesarias para crear el árbol sintáctico indicando los argumentos de estas funciones.

   ecuacion → *id* = *expr*
   expr → *expr* + *expr* | - *expr* | (*expr*) | **id** | **num**

   Indica para la producción de la asignación cómo se hace la comprobación semántica de tipos.

3. Dada la grámatica que genera sentencias de control del tipo `if-then` y de escritura `write`, añade los correspondientes operadores en notación de PCCTS para generar el árbol de análisis sintáctico, compactando la gramática para dejarla en notación EBNF.
   stm → **if** *relExpr* **then** *(stm)** **end** **;** | **write** **id** **;** | **id** = *aritExpr* **;**
   relExpr → **id** (< **id** | == **id**)
   aritExpr → *aritTerm* *aritExprP*
   aritExprP → ’ + ’ *aritTerm* *aritExprP* | ϵ
   aritTerm → *aritFactor* *aritTermP*
   aritTermP → ’ * ’ *aritFactor* *aritTermP* | ϵ
   aritFactor → **num** | **id**

   Dibuja la forma del árbol de análisis sintáctico que genera PCCTS para la entrada:

```
if a<b then
a=2*a+b;
end;
write a ;
```

4. Supongamos que desea generar el código intermedio para una sentencia del tipo `repeat-until` de la forma:

$$RepeatStm \rightarrow \textbf{repeat}\ (Statment)*\ \textbf{until}\ BoolExpr\ ;$$

   Indica el fragmento de código que habría que añadir a la función generar código de la práctica 3, implementada en PCCTS para la clase AST, con el fin de que contemple este tipo de instrucciones.

5. Escribe una gramática $LL(1)$ que genere el siguiente conjunto de declaraciones de variables en C correspondiente al siguiente ejemplo:

```
int a[10];
int x;
int c,b,d[3];
```