Ejercicios de Sintaxis

Nota: los ejercicios marcados con (*) al principio están sacados del libro de la cátedra
los ejercicios marcados con (°) al principio están basados en uno tomado en un final

1. (°) Dada la siguiente BNF de un sublenguaje de ANSI C indique en el cuadro por SI o NO si las sentencias son derivables de esta BNF, y en caso de serlo, si son ejecutables en ANSI C

```
<sentencia> → while ( <expresión> ) <sentenciaExpresión> |
              <sentenciaExpresión>
<sentenciaExpresión> → <expresión> ;
<expresión> → <expPrimaria> = <expPrimaria> | <expPrimaria>
<expPrimaria> → 'a' | 2
```

SENTENCIA	DERIVABLE	EJECUTABLE
`while ('a' = 2) 2 = 2;`	Si	No
`'a' = 2;`	Si	No
`while ('a') 2;`	Si	Si
`2;`	Si	Si
`'a' = 2 = 2;`	No	---

2. (°) Dada la siguiente BNF de un sublenguaje de ANSI C indique en el cuadro por SI o NO si las sentencias son derivables de esta BNF, y en caso de serlo, si son ejecutables en ANSI C

```
<sentencia> → do <sentExpresión> while (<expresión>); | <sentExpresión>
<sentExpresión> → <expresión> ;
<expresión> → <expPrimaria> | <expresión> + <expPrimaria>
<expPrimaria> → 'z' | 22 | 4.66
```

SENTENCIA	DERIVABLE	EJECUTABLE
`do while (22);`	No	---
`do 'z'; while (22+4.66+22);`	Si	Si
`22+'z';`	Si	Si
`do 22+22 while (22);`	No	---

3. Dado el siguiente fragmento de código ANSI C liste en la tabla los lexemas que reconoce el escáner e indique a que categoría pertenecen.

```
int i;
double v[10] , *p;
/* fin declaraciones, inicio sentencias */
for (i = 0; i < 10; i++) {
      p = una_funcion(i);
      v[i] = *p;
}
```

lexema	token	lexema	token	lexema	token
int	palabReservada	i	identificador	una_funcion	identificador
i	identificador	=	operador	(	operador
;	caracPuntuación	0	constante	i	identificador
double	palabReservada	;	caracPuntuación	)	operador
v	identificador	i	identificador	;	caracPuntuación
[	caracPuntuación	<	operador	v	identificador
10	constante	10	constante	[	operador
]	caracPuntuación	;	caracPuntuación	i	identificador
,	caracPuntuación	i	identificador	]	operador
*	caracPuntuación	++	operador	=	operador
p	identificador	)	caracPuntuación	*	operador
;	caracPuntuación	{	caracPuntuación	p	identificador
for	palabReservada	p	identificador	;	caracPuntuación
(	caracPuntuación	=	operador	}	caracPuntuación

4. (°) Sean las categorías léxicas: <palabraReservada>, <identificador>, <operador>, <constante>, <otro>. Analice el siguiente fragmento de código complete la tabla con los lexemas y su categoría correspondiente

```
void XX VOID) { a++26; }
```

LEXEMA	CATEGORÍA LÉXICA
void	<palabraReservada>
XX	<identificador>
VOID	<identificador>
)	<otro>
{	<otro>
a	<identificador>
++	<operador>
26	<constante>
;	<otro>
}	<otro>

5. (°) Sea el siguiente fuente en ANSI C:

```
#define a 10
b/*e*/c/**/da:a=b/***/*/10"\*a*/"
```

Escriba en la tabla de abajo los lexemas detectados por el escáner y a que categoría pertenecen, suponiendo que estas son: IDENTIFICADOR, CONSTANTE y OTRO

LEXEMA	CATEGORÍA LÉXICA
b	IDENTIFICADOR
c	IDENTIFICADOR
da	IDENTIFICADOR
:	OTRO
10	CONSTANTE
=	OTRO
b	IDENTIFICADOR
*	OTRO
/	OTRO
10	CONSTANTE
“*a*/”	OTRO (no es CONSTANTE)

6. (°) Sean las siguientes categorías léxicas:
   TOKEN → <palRes> | <ident> | <otro> | <constReal>
   Realice el análisis léxico de la siguiente función ANSI C:
   INT main printf) { ++; 2.3E-8; }

LEXEMA	CATEGORÍA LÉXICA
INT	<ident>
main	<ident>
printf	<ident>
)	<otro>
{	<otro>
++	<otro>
;	<otro>
2.3E-8	<constReal>
;	<otro>
}	<otro>

7. (°) Sea la siguiente sentencia compuesta ANSI C

```
{ int i=0; int c;      8
  for ( ; isdigit(c); i++) ;12
  ungetc(c, stdin); }  7
```

   Determine
   a) Cantidad de lexemas que hay DENTRO de la sentencia compuesta: **27**
   b) Cantidad de sentencias que hay DENTRO de esta sentencia compuesta: **2**

8. (°) Sean las categorías gramaticales (o sintácticas) **Expresiones**, **Declaraciones** y **Sentencias** de ANSI C. Indique que categoría, según el compilador, le corresponde a cada uno de los siguientes constructos:

CONSTRUCTO	CATEGORÍA
`{printf;}`	Sentencia
`main = scanf`	Expresión
`struct punto {double x; double y;} punto;`	Declaración
`int a, x(void);`	Declaración
`a+++++b;`	Sentencia
`While(1);`	Sentencia

9. (°) Indique el tipo de dato (en ANSI C) de cada una de las siguientes expresiones o escriba ERROR si no es una expresión

EXPRESIÓN	TIPO DE DATO
`-0.266`	`double`
`14L`	`long int`
`22.6 > 4`	`int`
`'a' + 22.F`	`float`
`12.8 && -12.8`	`int`

10. (°) Por cada cadena de la siguiente tabla indique a que CATEGORÍA DE CONSTANTE pertenece el lexema asociado. Las categorías son: entera decimal, entera hexadecimal, entera octal, real, carácter. Además escriba el TIPO DE DATO de cada constante. Si no es posible formar una constante de las categorías dadas, escriba ERROR.

Cadena	Categoría de Constante	Tipo de dato
`7f`	`ERROR`	`---`
`'\\'`	`carácter`	`int`
`10L`	`entera decimal`	`long int`
“C”	`ERROR`	`---`
`4.3e8`	`real`	`double`
`0127`	`entera octal`	`int`

11. (°) Marque con una cruz si los siguientes constructos ANSI C tiene errores semánticos, errores sintácticos, o no tiene error de compilación. Asuma que las funciones estándar están disponibles.

	Errores Semánticos	Errores Sintácticos	Sin Error
`{int main=0; printf(“%d\n”, main);}`			x
`{char a[5]; a[15]='A'+2;}`			x
`{int a=0,b; {while (a<10) {++a; b=b+2;}}`		x	
`{int p=5; for ( ;5; ) 8 = p-- ;}`	x		

12. Marque con una cruz si los siguientes constructos ANSI C tiene errores semánticos, errores sintácticos, o no tiene error de compilación. Asuma que las funciones estándar están disponibles.

	Errores Semánticos	Errores Sintácticos	Error Léxico	Sin Error
{char a='a'; for(;;) 'F'=a++;}	x			
{char a[10]; a[-2]='B'+2;}				x
{int x=0,y=1,z; if x<y z=x+y;}		x		
{int a=0; a @= 1;}			x	
{int i,j=5; void *p; p=&j; i=*p;}	x			

13. Marque con una cruz si los siguientes constructos ANSI C tiene errores semánticos, errores sintácticos, o no tiene error de compilación. Asuma que las funciones estándar están disponibles.

	Errores Semánticos	Errores Sintácticos	Error Léxico	Sin Error
{int 2z=0; 2z += 1;}			x	
{int a=7; while a-- printf("%d", a);}		x		
{char *a = malloc(5); a[8]='A'+2;}				x
{int a; float a; for(;;) a++;}	x			
{int v[5],*p; p=v; *v++=3; *p++=2;}	x			