Practicando BNF y EBNF

TENER EN CUENTA QUE:

Meta símbolos utilizados por		Significado
BNF	EBNF	
< >	< >	Definición de un elemento no terminal
::=	::=	Definición de una producción
|	(|)	Selección de una alternativa
|< p > < p1 >	{ }	Repetición
	*	Repetición de 0 o más veces
	+	Repetición de 1 o más veces
	[ ]	Opcional está presente o no lo está
Nota: p y p1 son producciones si		

📘 Ejercicio 1: Números enteros positivos

Descripción: Definí la gramática para describir números enteros positivos (sin signo), como `1`, `42`, `999`.

Con BNF:

```
G = ( N , T , S , P )
N = <numeroEntero>,<Digito>
T = 0,1,2,3,4,5,6,7,8,9
S = <numeroEntero>
P = {

<numeroEntero>::= <digito> | <digito> <numeroEntero>
<digito>::= 0|1|2|3|4|5|6|7|8|9

}
```

Con EBNF

```
G = ( N , T , S , P)
N = <numeroEntero>, <digito>
T = 0,1,2,3,4,5,6,7,8,9
S = <numeroEntero>
P = {

<numeroEntero> ::= {<digito>}+
// o tambien puede ser <numeroEntero> ::= <digito> {<digito>}*

<digto>::= 0|1|2|3|4|5|6|7|8|9

}
```

📘 Ejercicio 2: Numero entero (positivo o negativo)

Con BNF

```
G = ( N , T , S , P )
N = <numeroEntero>, <digito>,<numero>
T = 0,1,2,3,4,5,6,7,8,9, " - "
S = <numeroEntero>
P = {

<numeroEntero>::= "-" <numero> | <numero>
<numero>::= <digito> | <digito> <numero>
<digito> ::= 0|1|2|3|4|5|6|7|8|9

}
```

Con EBNF:

```
G = ( N , T , S , P )
N = <numeroEntero> , <digito>
T = 0,1,2,3,4,5,6,7,8,9,"-"
S = <numeroEntero>
P = {
```

```
<numeroEntero> ::= [ "-" ] {digito}+
<digito> ::= 0|1|2|3|4|5|6|7|8|9

}
```

📘Ejercicio 3 Identificadores simples

Descripción: Definí una gramática que represente un identificador válido (como en muchos lenguajes de programación), donde:

- comienza con una letra (a-z, A-Z)
- seguido de cero o más letras o dígitos

Nota: Un identificador no puede comenzar con " _ "

Con BNF

Nota: Este ejercicio no está mal en cuanto a logica, pero repite codigo. Abajo lo corrijo.

```
G = ( N , T , S , P)
N = <identificador>,<letra>,<digito>, <caracter>,<caracterNoEspecial>,
<caracterEspecial>
T = 0,1,2..9, "a".."z", "A".."Z", "_"
S = <identificador>
P = {

<identificador> ::= <letra> | <letra> <caracter>

<caracter> ::= <caracterNoEspecial> <caracter> | <caracterEspecial <caracter>
| <caracterEspecial> | <caracterNoEspecial>

<caracterEspecial> ::= "_" <caracterEspecial> | "_"

<caracterNoEspecial> ::= <letra> <catacterNoEspecial> | <digito>
<caracterNoEspecial> | <letra> | <digito>

<letra> ::= "a" | "b" | ... "Z" | "A" | "B" .. | "Z"
<digito>::= 0|1|2|3|4|5|6|7|8|9
```

```
}
```

Correccion: Gramatica mas legible

```
G = ( N , T , S , P)
N = <identificador> , <letra> , <digito> , <caracter>, <caracterEspecial>
T = 0,1,2..9, "a".."z", "A".."Z", "_"
S = <identificador>
P = {

<identificador> ::= <letra> | <letra> <caracter>

<caracter> := <letra> <caracter> | <digito> <caracter> | <caracterEspecial>
<caracter> | <letra> | <digito> | <caracterEspecial>

<letra> ::= "a" | "b" | .. | "z" | "A" | "B" .. | "Z"
<digito> ::= 0|1|2|3|4|5|6|7|8|9
<caracterEspecial> ::= "_"

}
```

Con EBNF:

```
G = ( N , T , S , P )
N = <identificador> , <digito>, <letra> , <caracterEspecial>, <catacter>
T = 0,1,2..9, "a".."z", "A".."Z", "_"
S = <identificador>
P = {

<identificador> ::= <letra> {caracter}*
<caracter> ::= ( <letra> | <digito> | <caracterEspecial>)
<letra> ::= "a" | "b" | .. | "z" | "A" | "B" .. | "Z"
<digito> ::= 0|1|2|3|4|5|6|7|8|9
<caracterEspecial> ::= "_"

}
```

📘 Ejercicio 4: Listas separadas por coma

Descripción: Definí una gramática para una lista de palabras (letras solamente) separadas por comas. Ejemplo:

- `hola`
- `hola,chau`
- `uno,dos,tres,cuatro`

Con BNF

```
G = ( N , T , S , P )
N = <palabra>, <letra>, <separador>
T = "a".."z", "A".."Z", ","
S = <palabra>
P = {

<palabra>::= <letra> <palabra> | <letra> <palabra> <separador> <palabra> |
<letra> <separador> <palabra> | <letra>

<letra> ::= "a"| "b"|..| "z"| "A" | "B" |... | "Z"
<separador>::= ","

}
```

Con EBNF

```
G = ( N , T , S , P )
N = <palabra>, <letra>, <separador>
T = "a".."z", "A".."Z", ","
S = <palabra>
P = {

<palabra>::= {letra}+ { <separador> {<letra>}+ }*

<letra>::= "a"| "b"|..| "z"| "A" | "B" |... | "Z"
<separador> ::= " , "
```

```
}
```