Ejercicio 4

Escriba un archivo calc4.y que implemente la gramática e imprima el valor de la expresión:

exp → numero | exp + exp | exp - exp | exp * exp | (exp)

Incluya las opciones:

```
%left '+' '-'
%left '*'
```

- ¿Es ambigua?
- ¿La precedencia ahora es correcta?
- Pruebe invertir las opciones para verificar el cambio

Ver carpeta Pregunta4

No es ambigua, ya que se eliminó todos los conflictos y ahora tenemos una buena precedencia gracias a %left '+' '-' y %left '*'

Ahora la precedencia es correcta.

Ya las probé

Ejercicio 5

Tomando como base el ejercicio 1 y calc.y genere un calc5.y y calc5.l donde integre el analizador lexicográfico.

Debe generar y usar el archivo calc5.tab.h
Tome como referencia:

```
number [0-9]+.?|[0-9]*.[0-9]+

%%
[ ] { /* skip blanks */ }
{number} { return NUMBER;
         }

\n|. {return yytext[0];}
```

Ver carpeta Pregunta5
calc5.tab.h se llama y.tab.h
correr:

```
yacc -d calc5.y
lex calc5.l
gcc lex.yy.c y.tab.c -ll -ly -o exec
./exec
```

Ejercicio 6

Ahora en los archivos calc6.y y calc6.l modifique lo que se necesario para que la gramática utilice el tipo de variable como `double` para permitir cálculos de coma flotante.

Para ello añada la definición:

```
#define YYSTYPE double
```

en el encabezado y haga que le resultado se imprima con dos decimales.

Ver carpeta Pregunta6

correr:

```
yacc -d calc6.y
lex calc6.l
gcc lex.yy.c y.tab.c -ll -ly -o exec
./exec
```