```
Lexico encontrado:


Variable:       var1

Line: 0, Chars(0, 4)


EqualSign:      =

Line: 0, Chars(5, 6)


Integer:        5

Line: 0, Chars(7, 8)


BreakLine:



Line: 1, Chars(8, 9)


Variable:       var2

Line: 1, Chars(0, 4)


EqualSign:      =

Line: 1, Chars(5, 6)


Float:  4.5

Line: 1, Chars(7, 10)


BreakLine:



Line: 2, Chars(10, 11)


Variable:       var3

Line: 3, Chars(0, 4)


EqualSign:      =
Line: 3, Chars(5, 6)


Variable:	var1

Line: 3, Chars(7, 11)


MultiOperator:  *

Line: 3, Chars(12, 13)


Variable:	var2

Line: 3, Chars(14, 18)


PlusOperator:   +

Line: 3, Chars(19, 20)


Integer:	10

Line: 3, Chars(21, 23)


BreakLine:


Line: 4, Chars(23, 24)


ReservedWord:   print

Line: 5, Chars(0, 5)


OpParenthesis:  (

Line: 5, Chars(5, 6)


StringDQ:	"Hola, el resultado es: "

Line: 5, Chars(6, 31)


PlusOperator:   +

Line: 5, Chars(32, 33)
```

```
Variable:       var3

Line: 5, Chars(34, 38)


CloParenthesis: )

Line: 5, Chars(38, 39)


BreakLine:


Line: 6, Chars(39, 40)


ReservedWord:   if

Line: 7, Chars(0, 2)


OpParenthesis:  (

Line: 7, Chars(3, 4)


Variable:       var3

Line: 7, Chars(4, 8)


GreaterThan:    >

Line: 7, Chars(9, 10)


Integer:        5

Line: 7, Chars(11, 12)


CloParenthesis: )

Line: 7, Chars(12, 13)


OpBrace:        {

Line: 7, Chars(14, 15)
```

```
BreakLine:


Line: 8, Chars(15, 16)


ReservedWord:   print

Line: 8, Chars(4, 9)


OpParenthesis:  (

Line: 8, Chars(9, 10)


StringSQ:       'Ingresa el n2: '

Line: 8, Chars(10, 27)


CloParenthesis: )

Line: 8, Chars(27, 28)


BreakLine:


Line: 9, Chars(28, 29)


ReservedWord:   readLn

Line: 9, Chars(4, 10)


OpParenthesis:  (

Line: 9, Chars(10, 11)


CloParenthesis: )

Line: 9, Chars(11, 12)


BreakLine:


Line: 10, Chars(12, 13)
```

```
CloBrace:      }

Line: 10, Chars(0, 1)


ReservedWord:   else

Line: 10, Chars(2, 6)


OpBrace:       {

Line: 10, Chars(7, 8)


BreakLine:


Line: 11, Chars(8, 9)


ReservedWord:   for

Line: 11, Chars(4, 7)


Variable:      n

Line: 11, Chars(8, 9)


OpParenthesis:  (

Line: 11, Chars(10, 11)


Integer:       5

Line: 11, Chars(11, 12)


CloParenthesis: )

Line: 11, Chars(12, 13)


OpBrace:       {

Line: 11, Chars(14, 15)
BreakLine:


Line: 12, Chars(15, 16)


ReservedWord:   print

Line: 12, Chars(8, 13)


OpParenthesis:  (

Line: 12, Chars(13, 14)


Variable:       n

Line: 12, Chars(14, 15)


CloParenthesis: )

Line: 12, Chars(15, 16)


BreakLine:


Line: 13, Chars(16, 17)


CloBrace:       }

Line: 13, Chars(4, 5)


BreakLine:


Line: 14, Chars(5, 6)


CloBrace:       }

Line: 14, Chars(0, 1)


BreakLine:
```

```
Line: 15, Chars(1, 2)


Variable:       nombre

Line: 16, Chars(0, 6)


EqualSign:      =

Line: 16, Chars(7, 8)


StringDQ:       "Hector Rdz"

Line: 16, Chars(9, 21)


BreakLine:


Line: 17, Chars(21, 22)


ReservedWord:   print

Line: 17, Chars(0, 5)


OpParenthesis:  (

Line: 17, Chars(5, 6)


StringDQ:       "Hola "

Line: 17, Chars(6, 13)


PlusOperator:   +

Line: 17, Chars(14, 15)


Variable:       nombre

Line: 17, Chars(16, 22)


PlusOperator:   +

Line: 17, Chars(23, 24)
StringDQ:       "!"

Line: 17, Chars(25, 28)


CloParenthesis: )

Line: 17, Chars(28, 29)


BreakLine:


Line: 17, Chars(29, 30)



¡Excelente! No hay errores de léxico
```