**Scanner**:

Scanner that gets the following tokens according to the following regular expressions

Tokens:

* number => e.g. 3257 integer numbers only involved
* sign => + | - | \* | / | ^
* identifier => x

Regular Expressions:

* identifier = x
* operation = + | - | \* | / | ^
* digit = [0-9]
* number = digit digit\* //one or more digit

State Machine for scanning tokens:

+ or – or \* or / or ^

x

[other]

digit

digit

other

[other] means not to be consumed

**Parser:**

Parser to calculate function result according to the following grammar

Grammar BNF

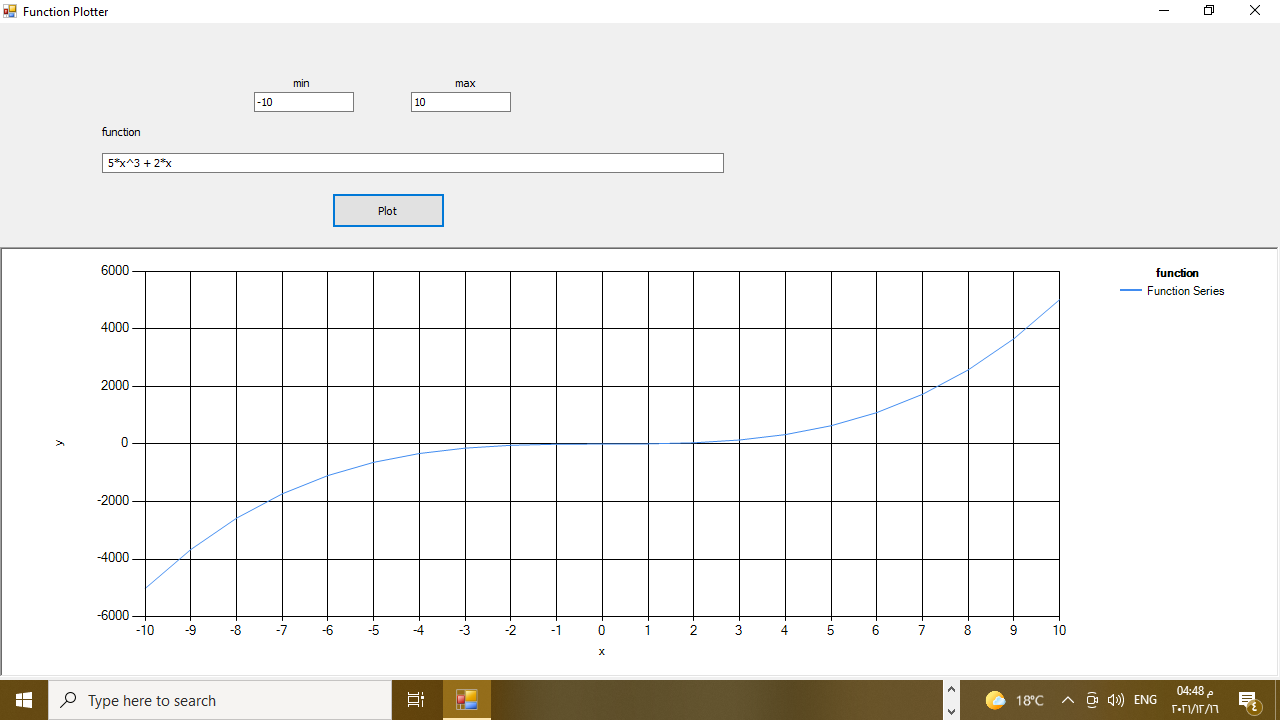
* addsubexp -> addsubexp addop muldivexp | muldivexp
* addop -> + | -
* muldivexp -> muldivexp mulop unaryexp | unaryexp
* mulop -> \* | /
* unaryexp -> + powerexp | - powerexp | powerexp
* powerexp -> exp powerop powerexp | exp
* powerop -> ^
* exp -> (addsubexp ) | identifier | number

Grammar EBNF

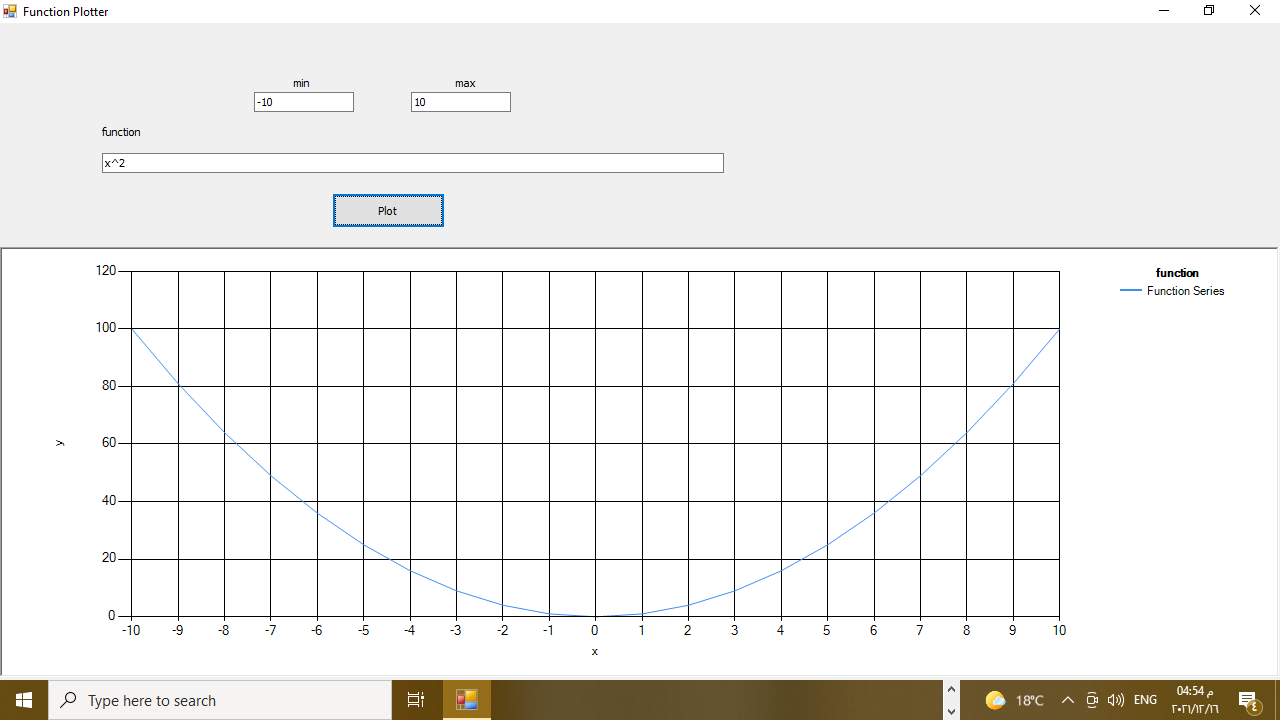
* addsubexp -> muldivexp { addop muldivexp }
* muldivexp -> unaryexp { mulop unaryexp }
* unaryexp -> + powerexp | - powerexp | powerexp
* powerexp -> { exp powerop } exp
* exp -> (addsubexp) | identifier | number

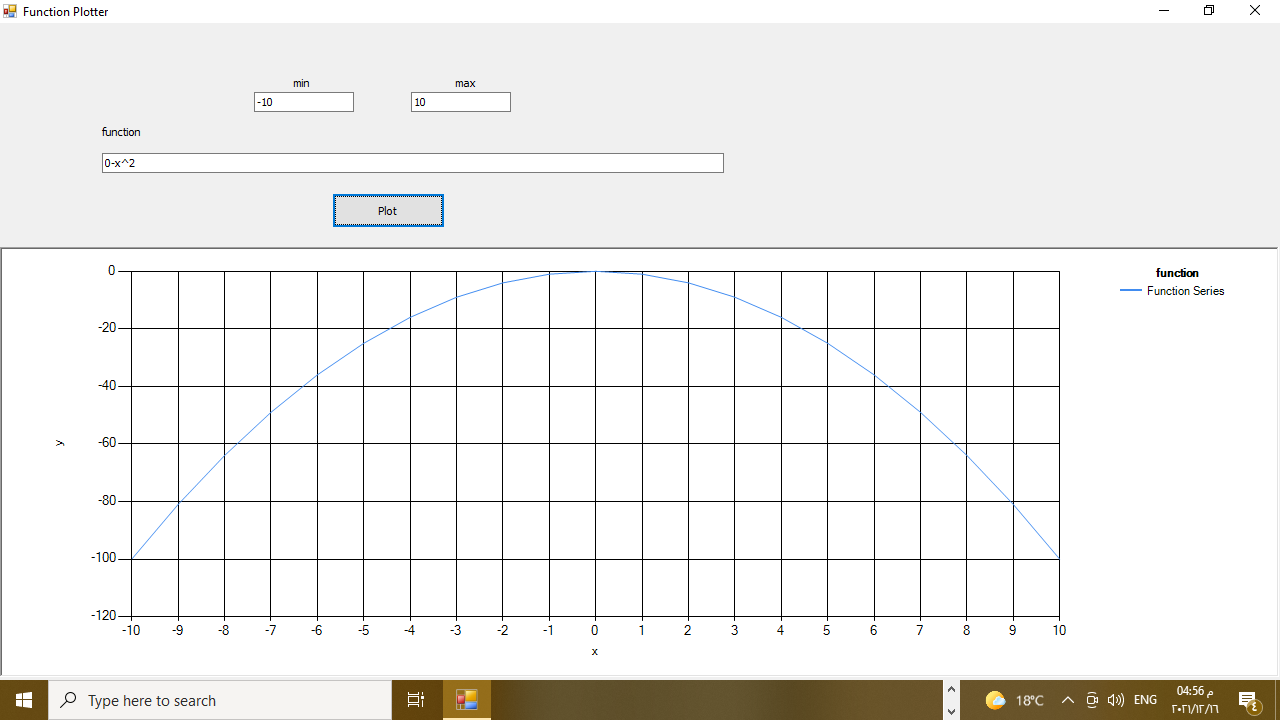
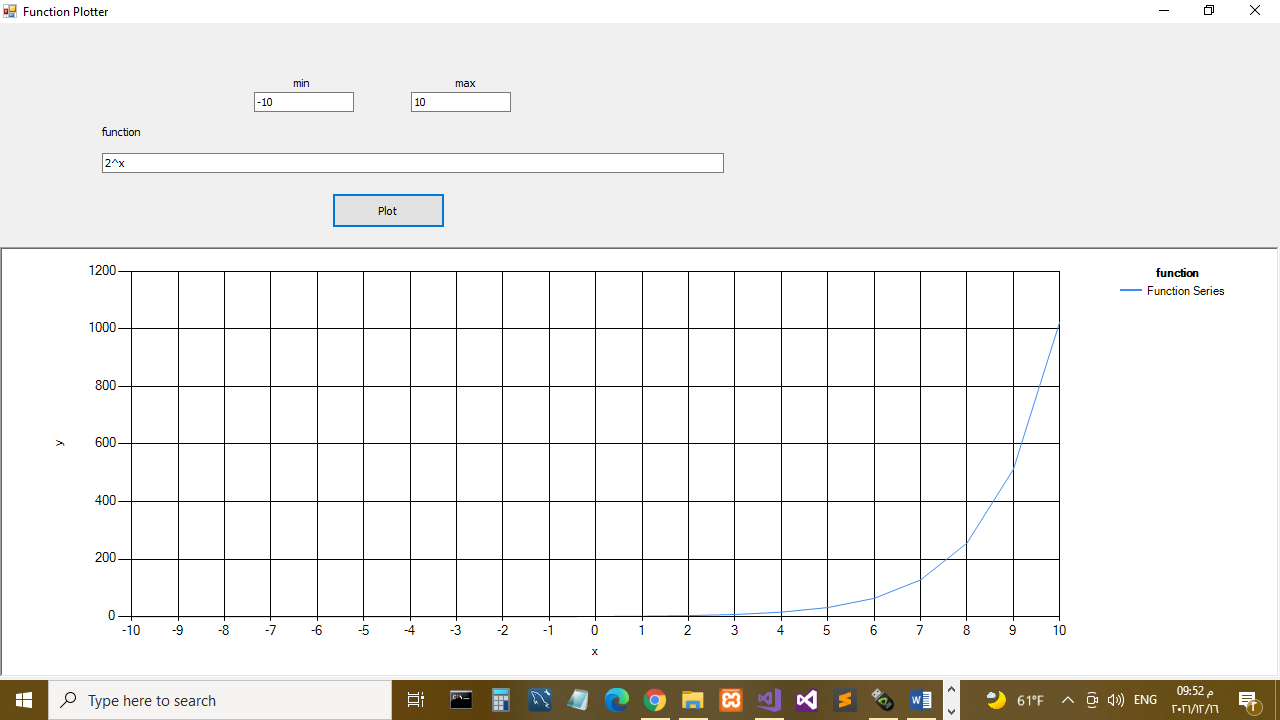
**working examples**

* 5\*x^3 + 2\*x



* x^2



* 0-x^2 works but –x^2 not handled yet
* 2^x