



Ain Shams University  
Faculty of Engineering  
Computers and Systems Engineering Department

## *Project 2* (Systems Software CSE422)

### Parser Project

section	code	name
2	1700684	شمس الدين عبدالرحمن علي علي
3	1701041	لؤي باسم نبيه عبد الواحد
3	1701008	كريم مجدي السيد حافظ محمد سليمان
4	1701136	محمد احمد ماهر احمد عبدالرحيم
5	1701730	يوسف عباس احمد عباس
5	1701671	ياسر احمد محمد احمد
5	1701665	وليد عبد العاطي حسن
5	1701588	نور الدين محمود مصباح كمال الحوت

# Parser Project

We used C# to build a GUI used to load any text file written by TINY language snippet code and then scanner it and convert grammar into EBNF form then parser it to generate a complete syntax tree.

## Tokens of the TINY language

TokenType	Value/Example
SEMICOLON	;
IF	if
THEN	then
END	end
REPEAT	repeat
UNTIL	until
IDENTIFIER	<ul style="list-style-type: none"><li>• x</li><li>• abc</li><li>• xyz</li></ul>
ASSIGN	:=
READ	read
WRITE	write
LESSTHAN	<
EQUAL	=
PLUS	+
MINUS	-
MULT	*
DIV	/
OPENBRACKET	(
CLOSEDBRACKET	)
NUMBER	<ul style="list-style-type: none"><li>• 12</li><li>• 289</li></ul>

We open Scanner.exe

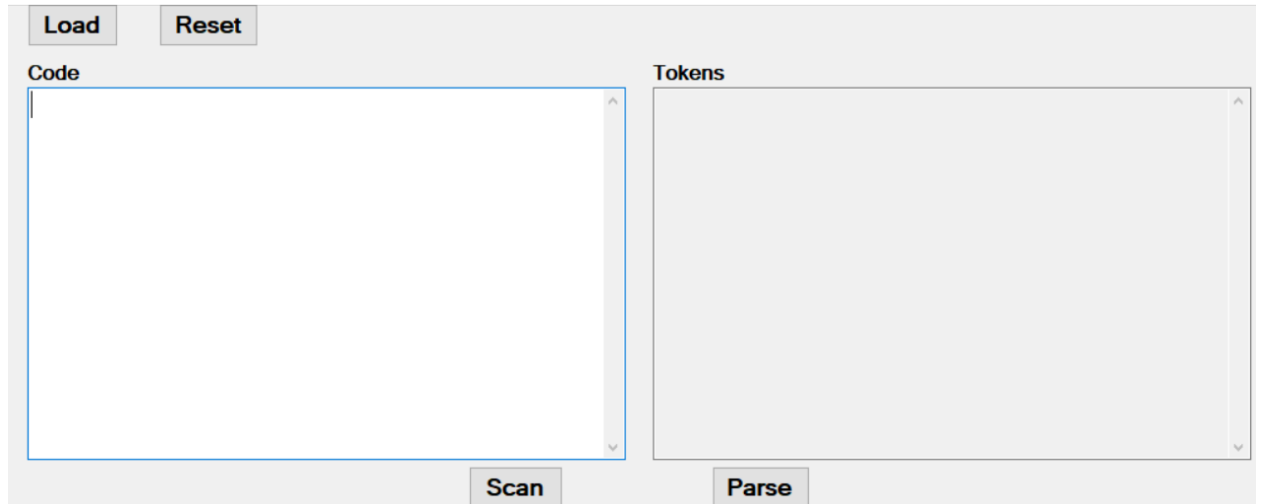


Figure 1 scanner application

Then we can add any text file we need by pressing on “Load” button.

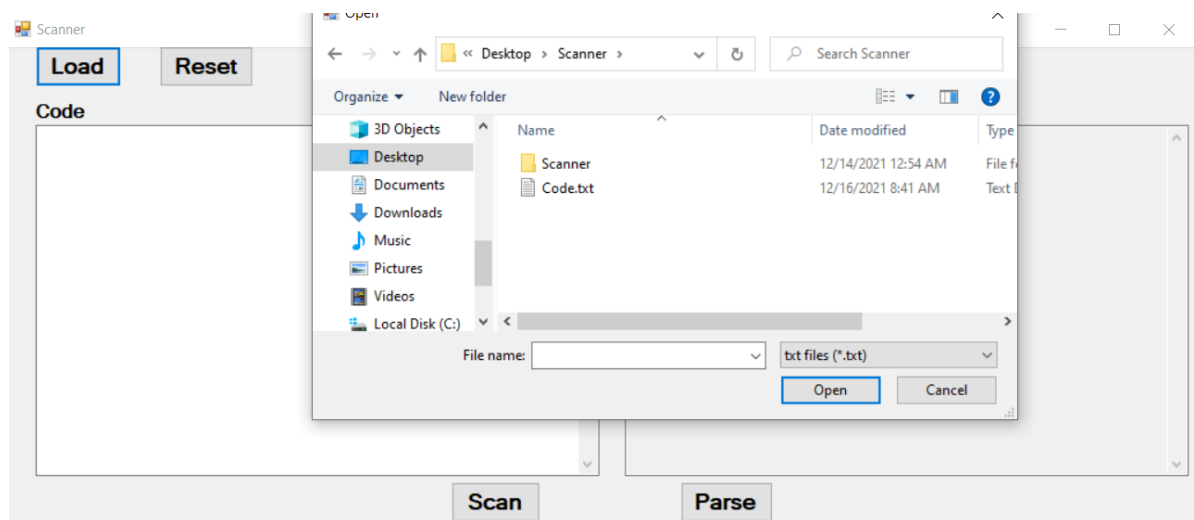


Figure 2 Load button

We will see that the context of file we loaded will appear in code textbox.



Figure 3 GUI after add text file

if we press scan to define each part of code

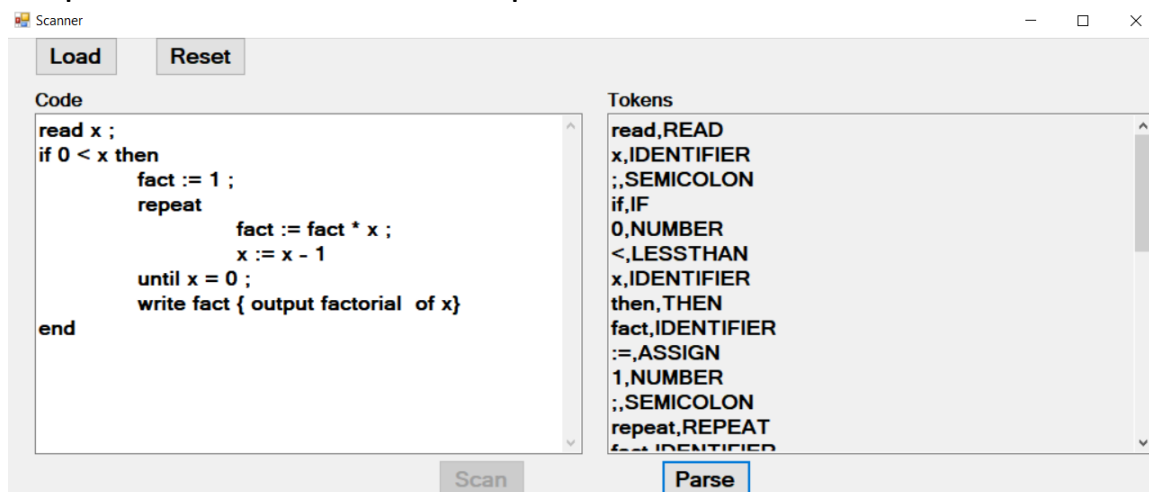


Figure 4 GUI after press scan button

if we press parse to draw syntax tree.

The screenshot shows a window titled "Scanner" with two main panels: "Code" and "Syntax Tree".

**Code Panel:**

```
read x ;
if 0 < x then
    fact := 1 ;
    repeat
        fact := fact * x ;
        x := x - 1
    until x = 0 ;
    write fact { output factorial of x }
end
```

**Syntax Tree Panel:**

The syntax tree is a hierarchical structure representing the code. The root node is "start", which branches into "READ(x)". "READ(x)" branches into "IF". "IF" branches into "op(<)" and "x". "op(<)" branches into "(0)". "(0)" branches into "(x)". "(x)" branches into "ASSIGN(fact)". "ASSIGN(fact)" branches into "NUMBER". "NUMBER" branches into "(1)". "(1)" branches into "REPEAT". "REPEAT" branches into "ASSIGN(fact)". "ASSIGN(fact)" branches into "op(\*)". "op(\*)" branches into "(fact)". "(fact)" branches into "(x)". "(x)" branches into "ASSIGN(x)". "ASSIGN(x)" branches into "op(-)". "op(-)" branches into "(x)". "(x)" branches into "(1)". "(1)" branches into "op(e)".

Buttons at the bottom: "Load", "Reset", "Scan", "Parse".

## For Ex2

Scan

The screenshot shows a window titled "Scanner" with two main panels: "Code" and "Tokens".

**Code Panel:**

```
read x ;
if x < 4 then
    write xyz ;
end
```

**Tokens Panel:**

```
read,READ
x,IDENTIFIER
;,SEMICOLON
if,IF
x,IDENTIFIER
<,LESSTHAN
4,NUMBER
then,THEN
write,WRITE
xyz,IDENTIFIER
;,SEMICOLON
end,END
```

Buttons at the bottom: "Load", "Reset", "Scan", "Parse".

Parse

The screenshot shows a window titled "Scanner" with two main panels: "Code" and "Syntax Tree".

**Code Panel:**

```
read x ;
if x < 4 then
    write xyz ;
end
```

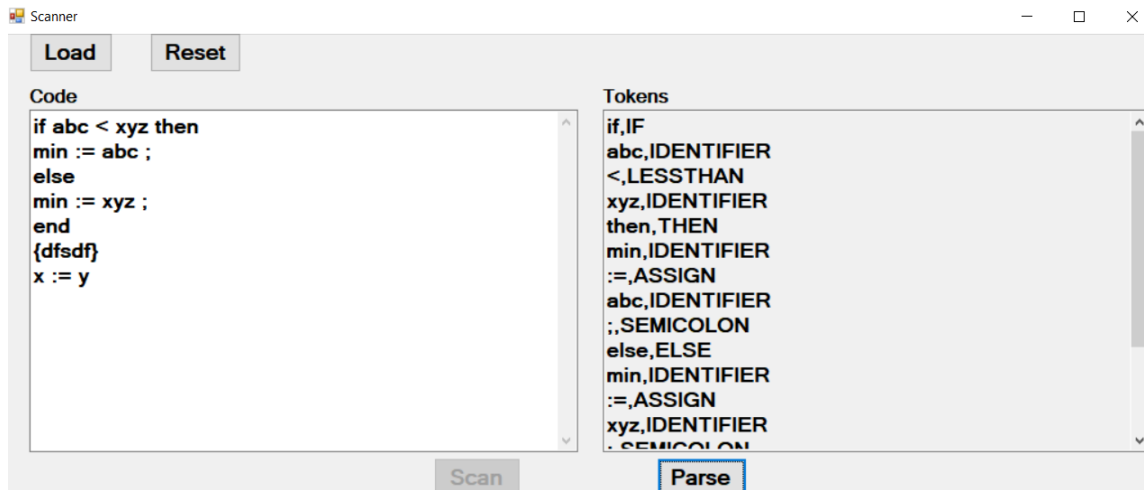
**Syntax Tree Panel:**

The syntax tree is a hierarchical structure representing the code. The root node is "start", which branches into "READ(x)". "READ(x)" branches into "IF". "IF" branches into "op(<)" and "x". "op(<)" branches into "(x)". "(x)" branches into "(4)". "(4)" branches into "WRITE". "WRITE" branches into "IDENTIFIER". "IDENTIFIER" branches into "(xyz)".

Buttons at the bottom: "Load", "Reset", "Scan", "Parse".

## For Ex3

### Scan



The Scanner window displays the following code in the 'Code' pane:

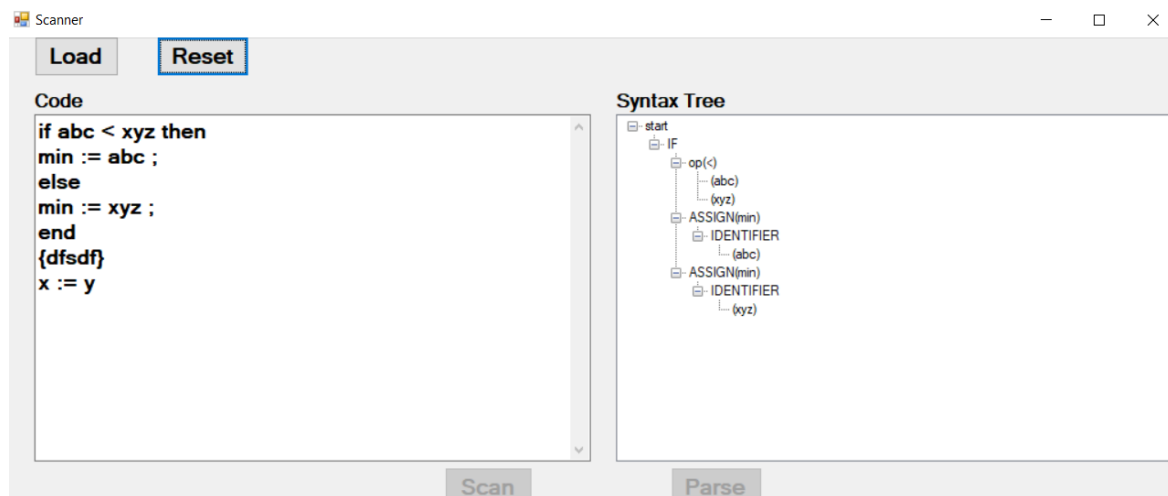
```
if abc < xyz then
min := abc ;
else
min := xyz ;
end
{dfsdf}
x := y
```

The 'Tokens' pane shows the following list of tokens:

```
if, IF
abc, IDENTIFIER
<, LESS THAN
xyz, IDENTIFIER
then, THEN
min, IDENTIFIER
:=, ASSIGN
abc, IDENTIFIER
;, SEMICOLON
else, ELSE
min, IDENTIFIER
:=, ASSIGN
xyz, IDENTIFIER
;, SEMICOLON
```

Buttons at the bottom include 'Load', 'Reset', 'Scan', and 'Parse'.

### Parse



The Parser window displays the same code in the 'Code' pane as the Scanner window.

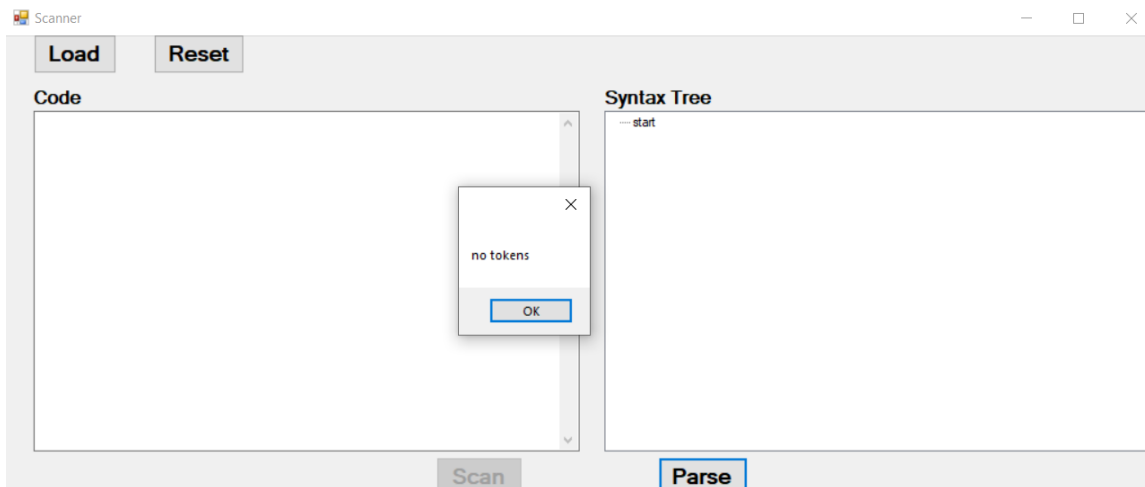
The 'Syntax Tree' pane shows the following tree structure:

```
graph TD
    start --> IF
    IF --> op["op(<)"]
    IF --> ASSIGN1["ASSIGN(min)"]
    IF --> ASSIGN2["ASSIGN(min)"]
    op --> abc1["(abc)"]
    op --> xyz1["(xyz)"]
    ASSIGN1 --> IDENT1["IDENTIFIER"]
    IDENT1 --> abc2["(abc)"]
    ASSIGN2 --> IDENT2["IDENTIFIER"]
    IDENT2 --> xyz2["(xyz)"]
```

Buttons at the bottom include 'Load', 'Reset', 'Scan', and 'Parse'.

## Check Errors

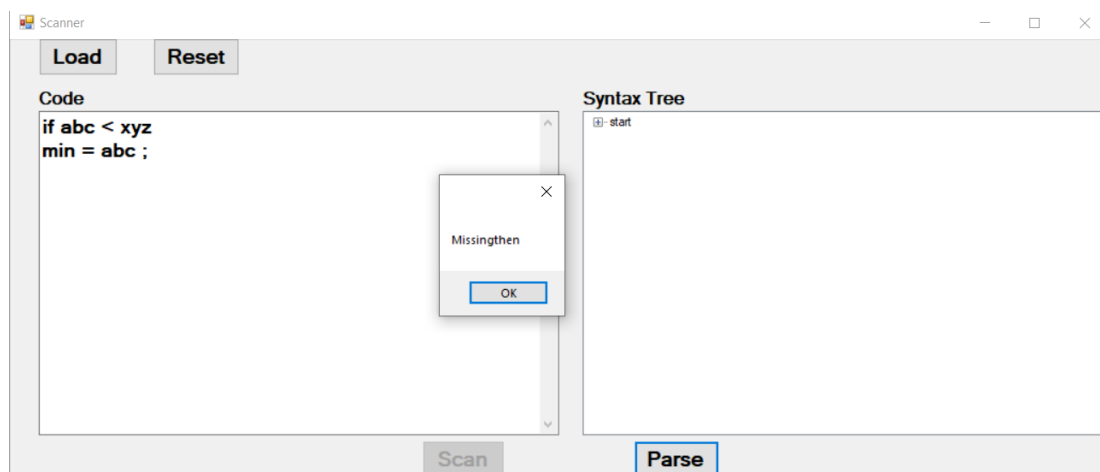
- 1- For no code and press parse  
So the program will display a message “No tokens”

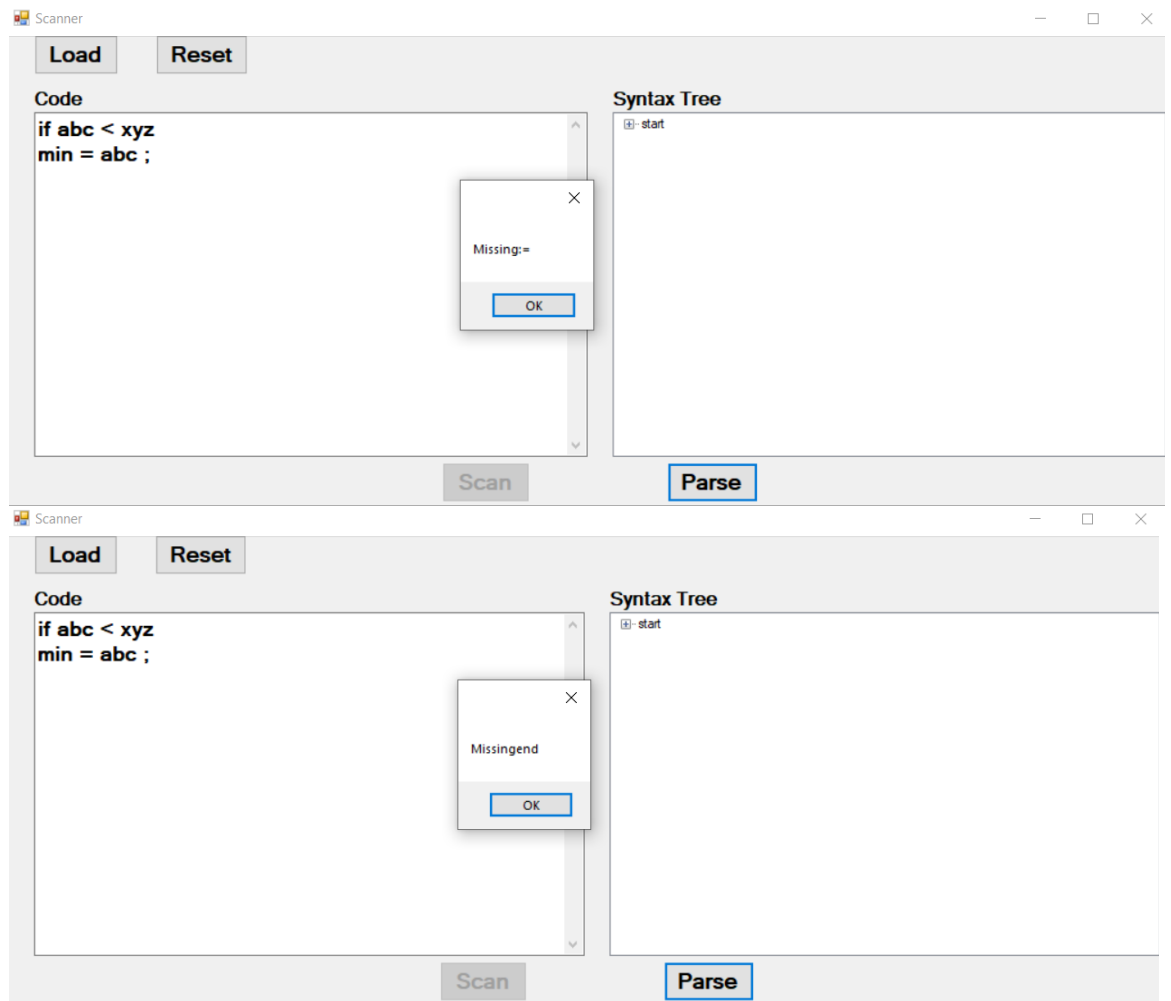


- 2- Check the syntax error of the tiny language  
So, with using this code

if abc < xyz  
min = abc ;

- here in this code as we studied that before each = we need to put “:”.
- for each if condition we need to put “then” and “end” at the end of the code of if.





## Drive Link

[https://drive.google.com/file/d/1U\\_xVVqNh17VbTpuwpQTJJO4m4yISFi5T/view?usp=sharing](https://drive.google.com/file/d/1U_xVVqNh17VbTpuwpQTJJO4m4yISFi5T/view?usp=sharing)