Mini Project 2
Parser

I. Project Description:

- Given the TINY grammar rules you should implement the TINY parser using recursive descent method.
- You will need to convert grammar into EBNF form.
- The output will be a complete syntax tree of the input source program

II. Inputs :

- List of (tokenvalue, tokentype)
  Example:
  x ,IDENTIFIER
  :=, ASSIGN
  4,NUMBER

- The input list should follow the same syntax as mentioned in the previous example **tokenvalue , tokentype** (Spaces are allowed between them)

- Input list can be input through GUI textbox or by loading a text file

- **List of token types in tiny language**

TokenType	Value/Example
SEMICOLON	;
IF	if
THEN	then
END	end
REPEAT	repeat
UNTIL	until
IDENTIFIER	• x • abc • xyz
ASSIGN	:=
READ	read
WRITE	write
LESSTHAN	<
EQUAL	=
PLUS	+
MINUS	-
MULT	*
DIV	/
OPENBRACKET	(
CLOSEDBRACKET	)
NUMBER	• 12 • 289

III. Output:

1. State whether the statements are accepted by TINY language or not
2. Draw Syntax tree on a GUI based application
3. IF you do not support GUI (and will lose GUI marks) you can output recognized structures by the TINY language parser into a file or on the console screen (like drawing the syntax tree by describing it using statement names)

INPUT

IV. Example

```
read,READ
x, IDENTIFIER
;,SEMICOLON
if,IF
0,NUMBER
<,LESSTHAN
x,IDENTIFIER
then,THEN
fact, IDENTIFIER
:=,ASSIGN
;,SEMICOLON
repeat,REPEAT
fact, IDENTIFIER
:=,ASSIGN
fact, IDENTIFIER
*,MULT
x, IDENTIFIER
;,SEMICOLON
x, IDENTIFIER
:=,ASSIGN
x, IDENTIFIER
-,MINUS
1,NUMBER
until,UNTIL
x, IDENTIFIER
=,EQUAL
0,NUMBER
;,SEMICOLON
write,WRITE
fact, IDENTIFIER
end,END
```

OUTPUT

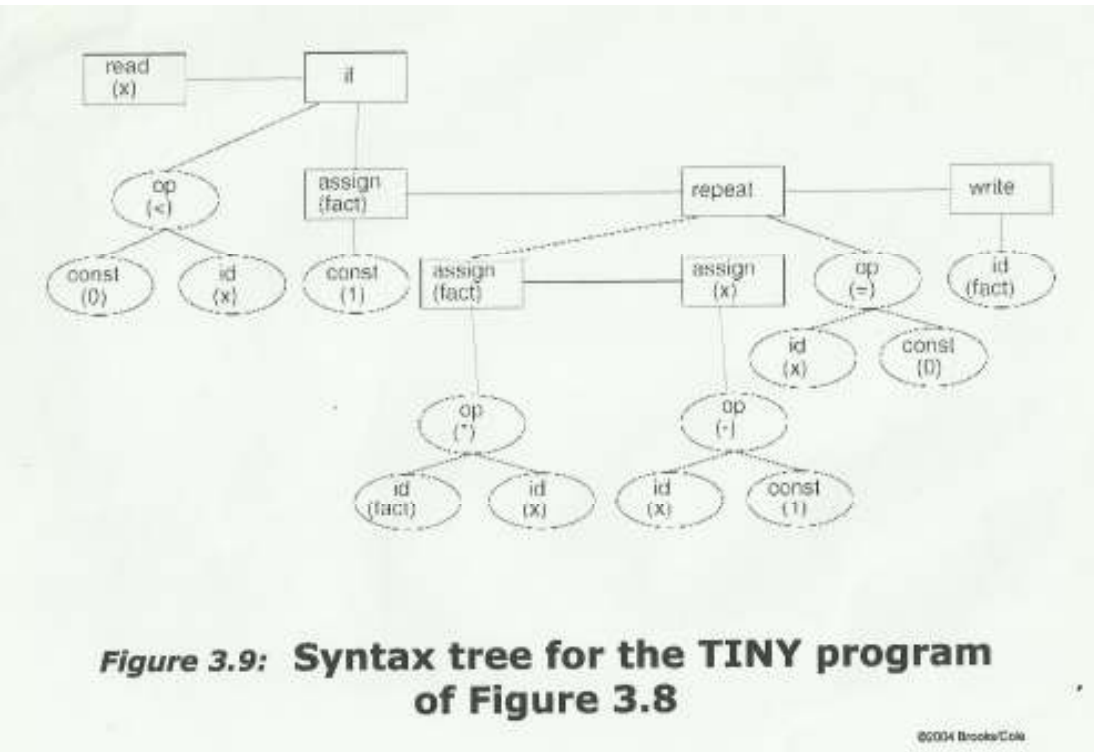

Figure 3.9: **Syntax tree for the TINY program of Figure 3.8**

Original Code Just for understanding the example

```
{ Sample program in TINY language – computes factorial
}
 read x;  {input an integer }
 if  0 < x  then    { don't compute if x <= 0 }
   fact  := 1;
   repeat
     fact  := fact *  x;
      x  := x  -  1
   until  x  =  0;
   write  fact   { output  factorial of x }
 end
```

V. Bonus

- Any error handling like if the user is requested to choose a file to parse then he chooses nothing and press OK (error) or if user enter an invalid file name (error) or any other error based on your program design.

VI. Deliverables

- Document delivered on lms by one of team members listing all group names. The document should include:
  - Working GUI Application as exe
  - Screen shots of examples worked on the gui
- **We will have a delivery by discussion after lms delivery : time is to be decided**

VII. Team:

Same teams as in scanner projects

VIII. Other Notes:

1- **You MUST commit to the same token types mentioned in the table with the same spelling and case sensitivity if needed.**
2- You MUST deliver a Desktop application executable.
3- You MUST provide a GUI Layer.
4- Your application must be able to run on new code without the need of reopening it.
5- **Due date : Thursday 30/12/2021**