SYSTEM SOFTWARE

Faculty of Engineering
Ain Shams University
Computer department

SUBMITTED BY:
Pierre Nabil (16E0056) Sec(1)
Girgis Michael (1600446) Sec(1)
John Bahaa (1600459) Sec(1)
Hazem Mohammed(1600469)Sec(1)

Pierre Nabil

Scanner and Parser Project

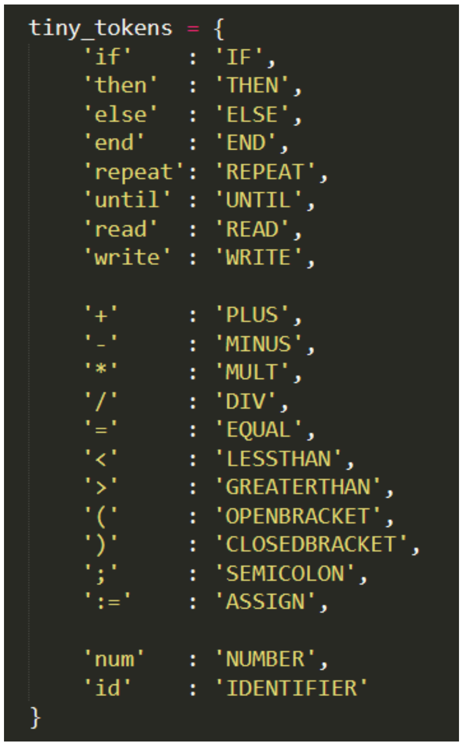
Table of Contents

Project 1: Scanner:

supported tokens are written in the shown figure.

They are written in the following format:

token_text : token_type



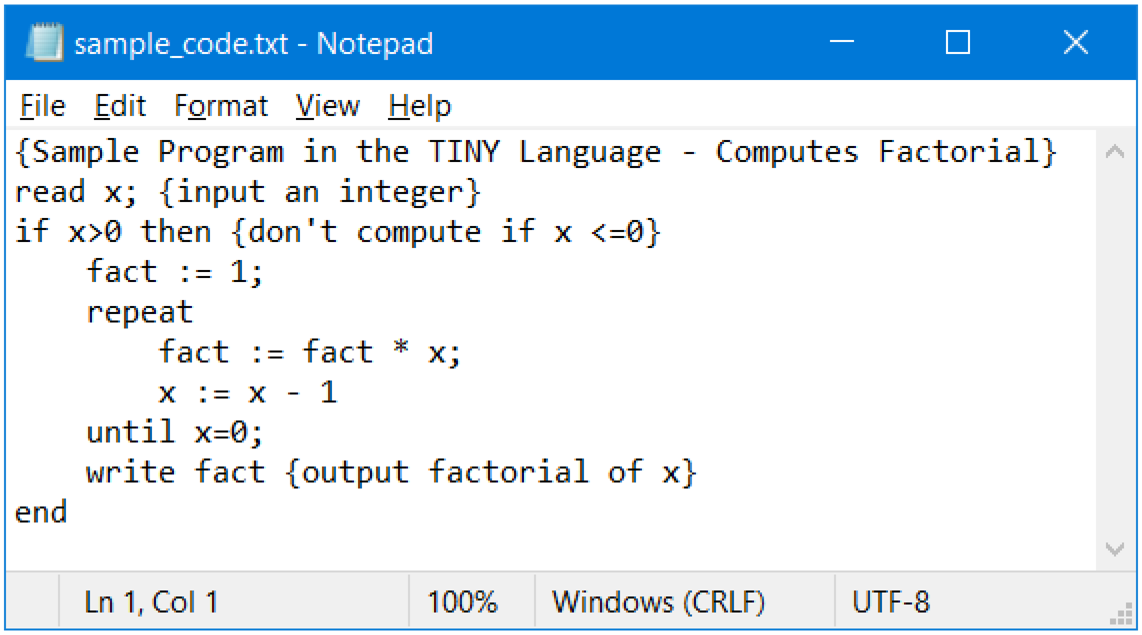
```
tiny_tokens = {
    'if'    : 'IF',
    'then'  : 'THEN',
    'else'  : 'ELSE',
    'end'   : 'END',
    'repeat': 'REPEAT',
    'until' : 'UNTIL',
    'read'  : 'READ',
    'write' : 'WRITE',

    '+'     : 'PLUS',
    '-'     : 'MINUS',
    '*'     : 'MULT',
    '/'     : 'DIV',
    '='     : 'EQUAL',
    '<'     : 'LESSTHAN',
    '>'     : 'GREATERTHAN',
    '('     : 'OPENBRACKET',
    ')'     : 'CLOSEDBRACKET',
    ';'     : 'SEMICOLON',
    ':='    : 'ASSIGN',

    'num'   : 'NUMBER',
    'id'    : 'IDENTIFIER'
}
```

Bonus Features:

- The Scanner supports extra tokens:
    - ELSE
    - GREATERTHAN
- The Scanner can read all identifier names as in the C++ language:
    - Must start with a letter or underscore
    - Can have any number of characters
    - Can include digits, letters and underscores
- The Scanner supports numbers with decimal notation and/or scientific notation.
    - Number can start with a digit or decimal point
    - Number can be written in scientific Notation (ex: 1e10 or 5.3E-5)

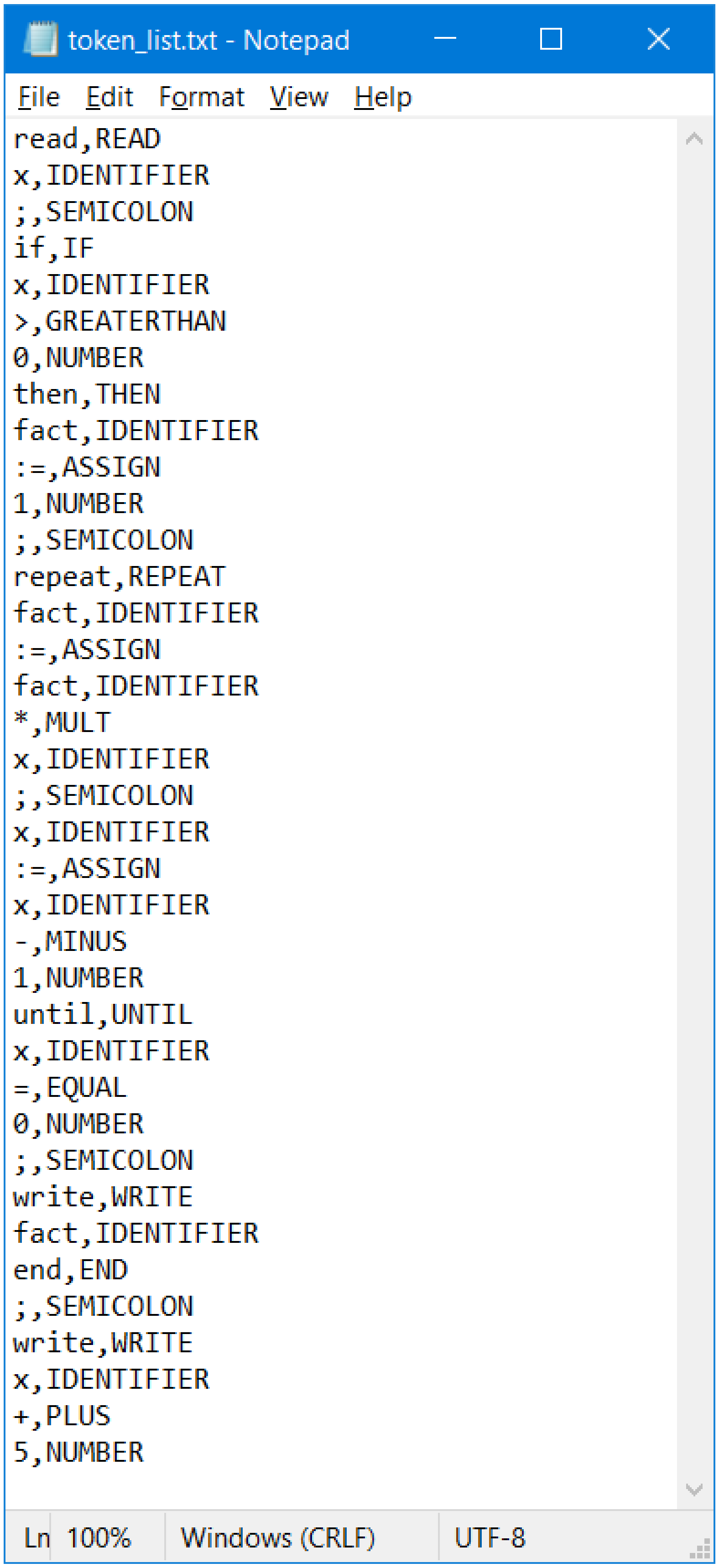
Example:

Input File:



```
{Sample Program in the TINY Language - Computes Factorial}
read x; {input an integer}
if x>0 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
```

Output File:



token_list.txt - Notepad

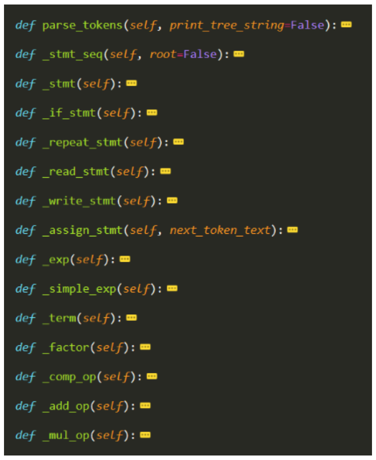
File Edit Format View Help

```
read,READ
x,IDENTIFIER
;,SEMICOLON
if,IF
x,IDENTIFIER
>,GREATERTHAN
0,NUMBER
then,THEN
fact,IDENTIFIER
:=,ASSIGN
1,NUMBER
;,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
;,SEMICOLON
write,WRITE
x,IDENTIFIER
+,PLUS
5,NUMBER
```

Ln 100% Windows (CRLF) UTF-8

Project 2: Parser:

supported list of non-terminals is shown in the figure.



```
def parse_tokens(self, print_tree_string=False):
def _stmt_seq(self, root=False):
def _stmt(self):
def _if_stmt(self):
def _repeat_stmt(self):
def _read_stmt(self):
def _write_stmt(self):
def _assign_stmt(self, next_token_text):
def _exp(self):
def _simple_exp(self):
def _term(self):
def _factor(self):
def _comp_op(self):
def _add_op(self):
def _mul_op(self):
```

Bonus Features:

- The Parser can draw the Parse Tree and/or the Syntax Tree of the scanned Tokens.
    - Not just the Syntax Tree
- The Parser uses the full TINY language Grammar

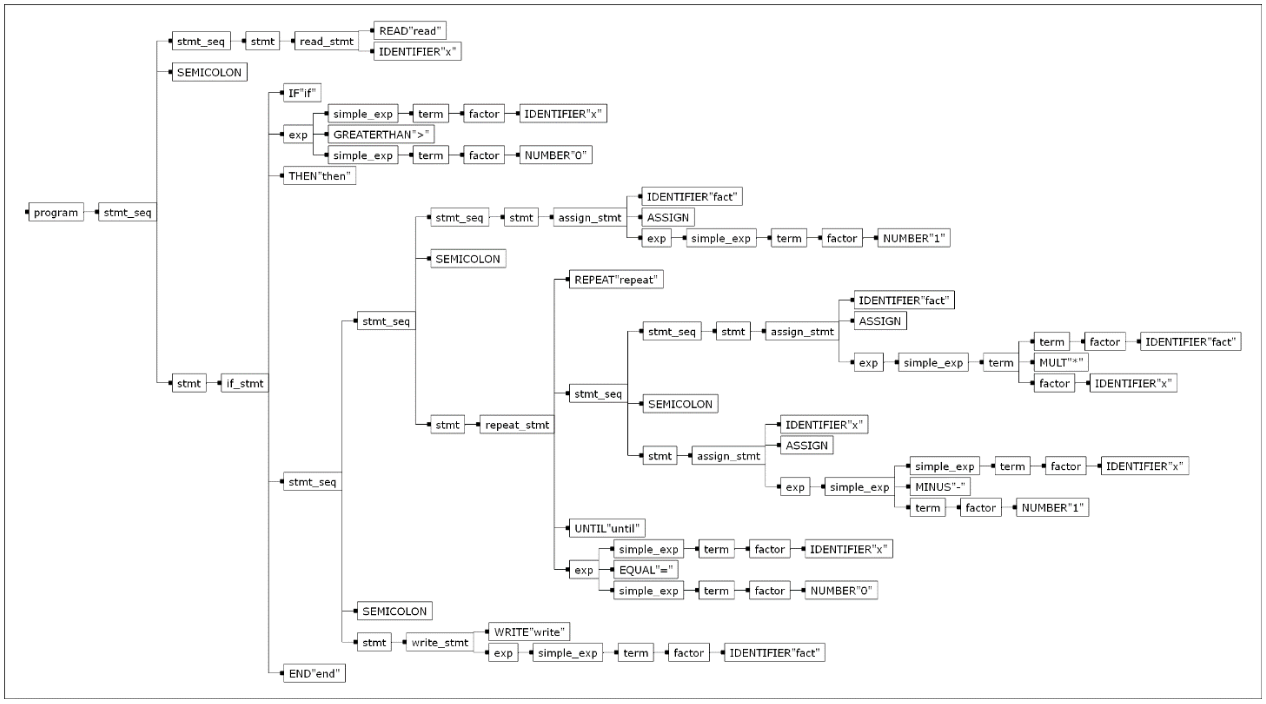
Example:

Input File:

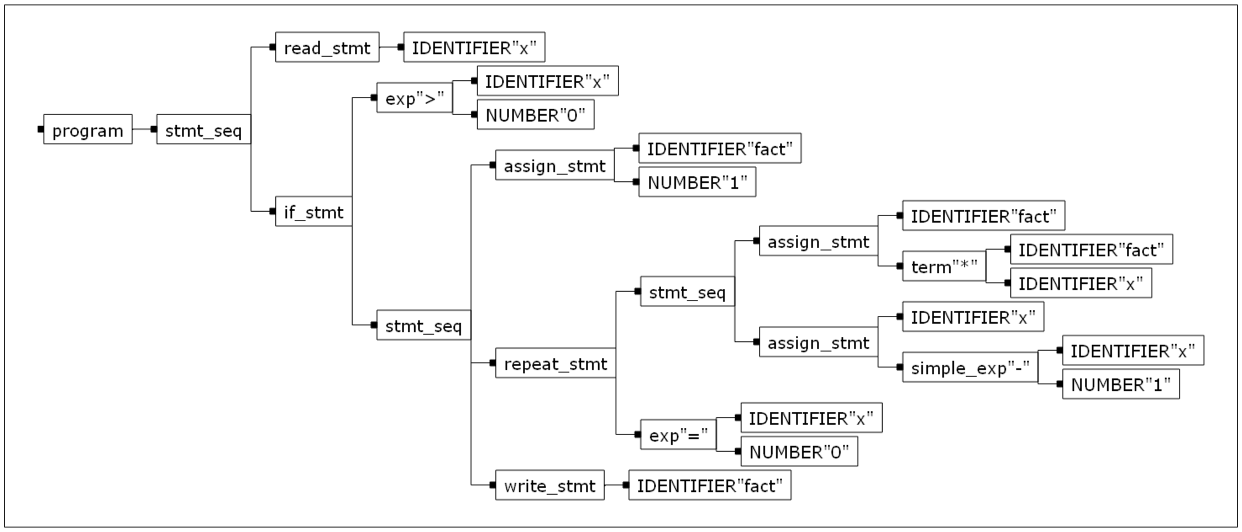
The Output File of the Scanner

Output File: (2 images)

Parse Tree:



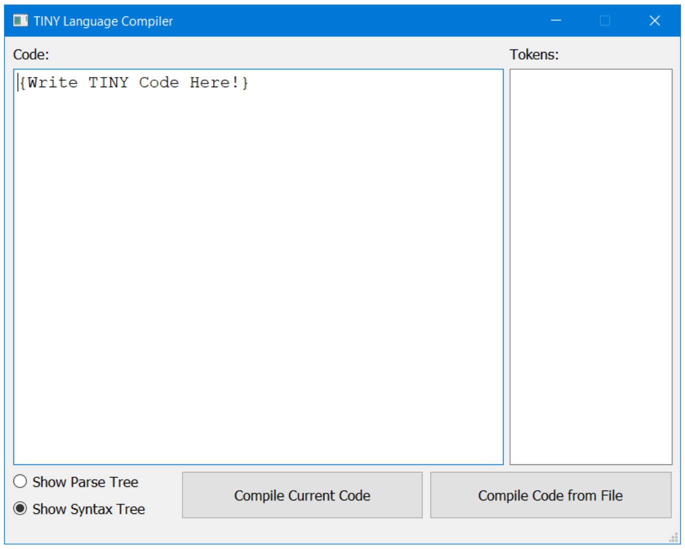
Syntax Tree:


program
stmt_seq
read_stmt
IDENTIFIER"x"
if_stmt
exp">"
IDENTIFIER"x"
NUMBER"0"
stmt_seq
assign_stmt
IDENTIFIER"fact"
NUMBER"1"
repeat_stmt
stmt_seq
assign_stmt
IDENTIFIER"fact"
term"*"
IDENTIFIER"fact"
IDENTIFIER"x"
assign_stmt
IDENTIFIER"x"
simple_exp"-"
IDENTIFIER"x"
NUMBER"1"
exp"="
IDENTIFIER"x"
NUMBER"0"
write_stmt
IDENTIFIER"fact"

GUI:

We built a GUI that takes as input TINY language code as input text in the left textbox or in a ".txt" file.

This GUI uses our scanner and parser projects to output the Syntax Tree of Parse Tree directly from the code, while showing the found tokens in the right textbox.



```
{Write TINY Code Here!}
```

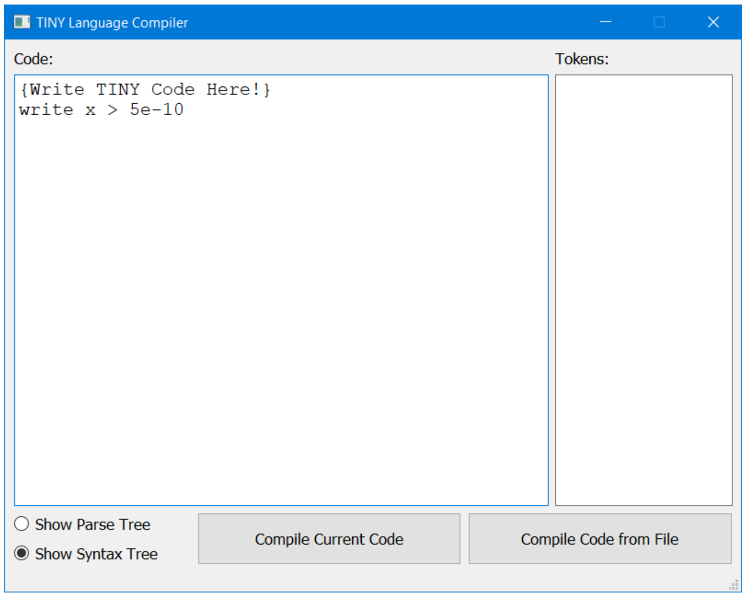
● Show Syntax Tree

Bonus Features:

- Doesn't allow the user to use an input file that is not a ".txt" file.
- Reports any Parsing Errors as error messages.
- Shows the Syntax and Parse Trees in an Interactive window.

Example:

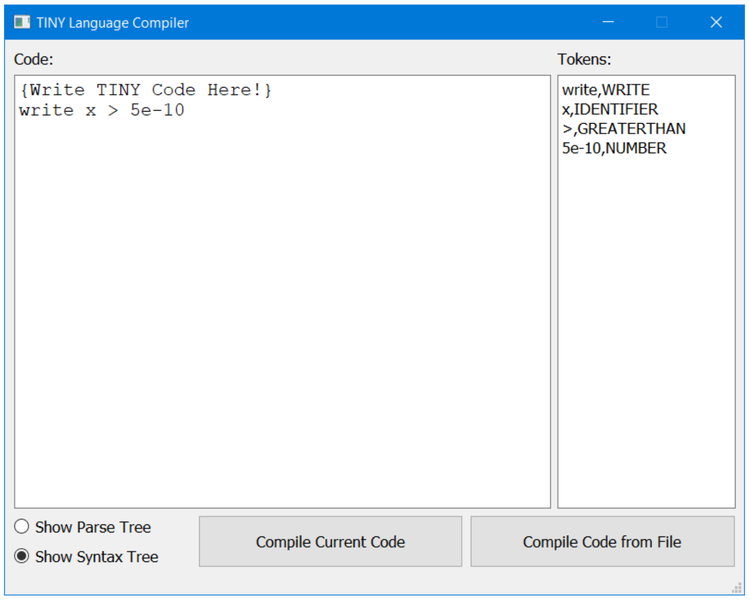
Startup window:



```
{Write TINY Code Here!}
write x > 5e-10
```

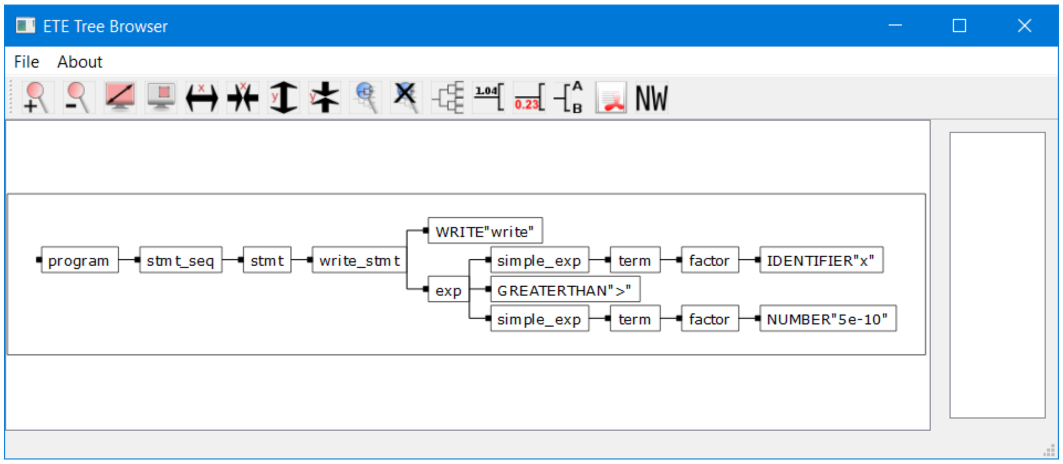
● Show Syntax Tree

Select “Show Parse Tree” then click “Compile Current Code”:

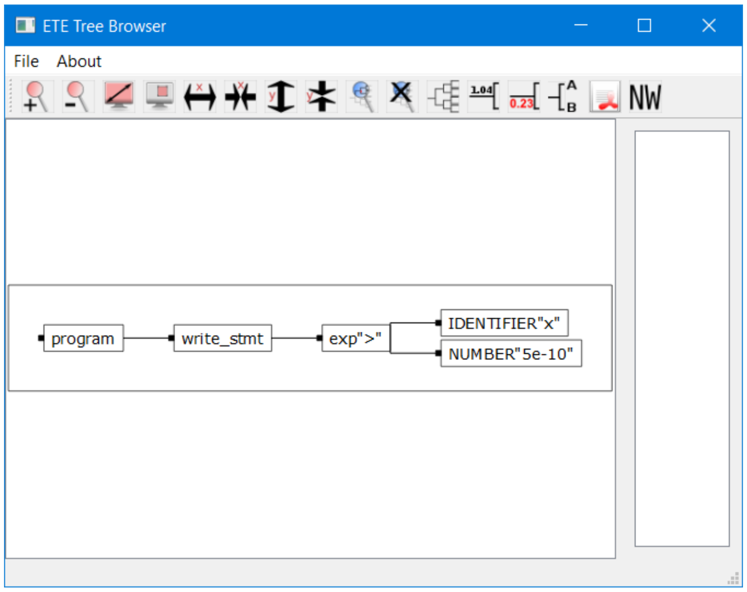


```
{Write TINY Code Here!}
write x > 5e-10
```

```
write,WRITE
x,IDENTIFIER
>,GREATERTHAN
5e-10,NUMBER
```

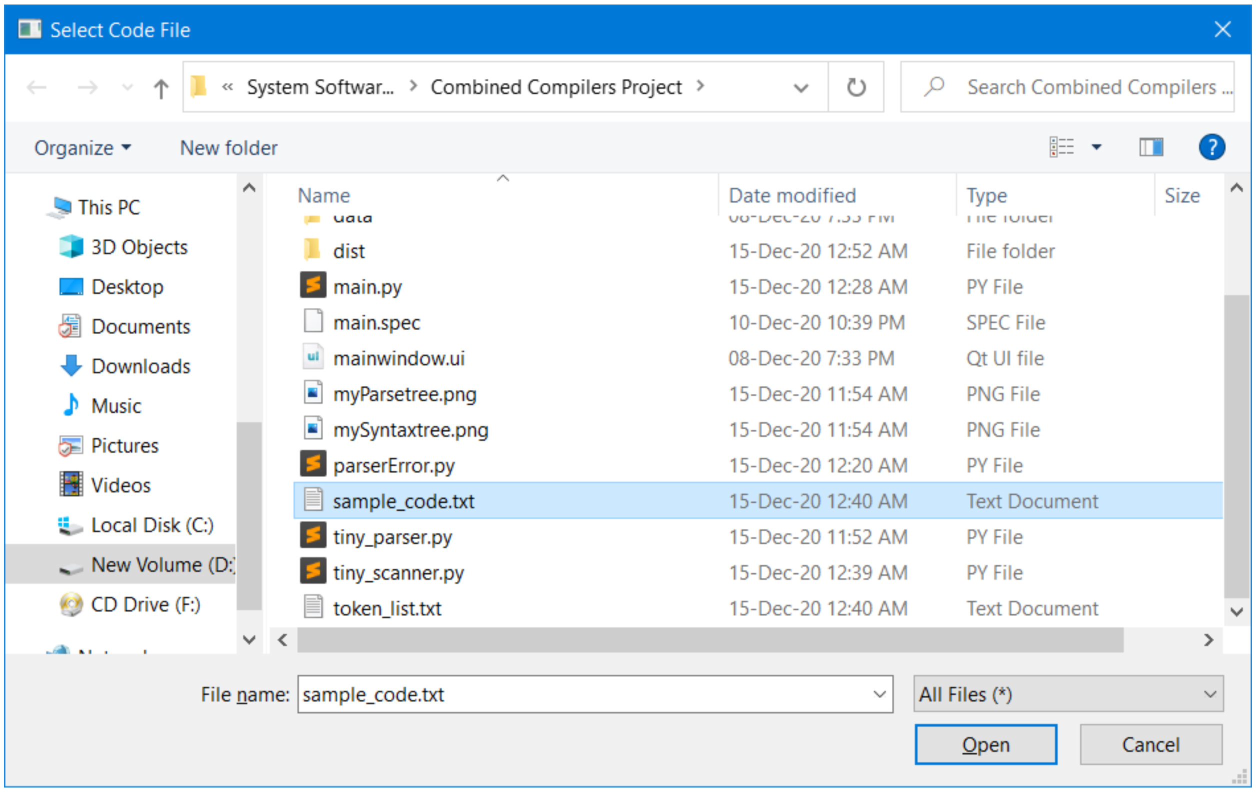


Select “Show Syntax Tree” then click “Compile Current Code”:



Select “Show Syntax Tree” then click “Compile Code From File”:

Note: Same steps for Parse Tree.



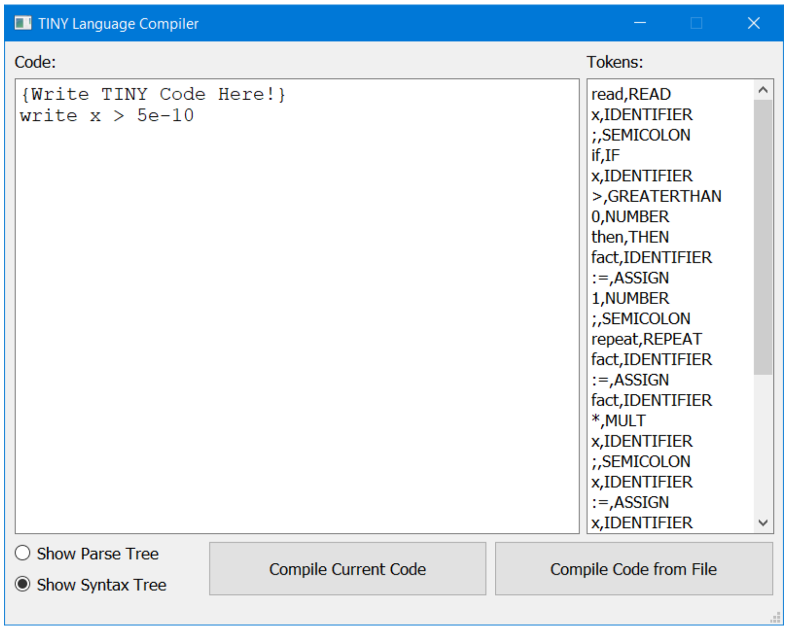
Name	Date modified	Type	Size
data	08-Dec-20 7:33 PM	File folder	
dist	15-Dec-20 12:52 AM	File folder	
main.py	15-Dec-20 12:28 AM	PY File	
main.spec	10-Dec-20 10:39 PM	SPEC File	
mainwindow.ui	08-Dec-20 7:33 PM	Qt UI file	
myParsetree.png	15-Dec-20 11:54 AM	PNG File	
mySyntaxtree.png	15-Dec-20 11:54 AM	PNG File	
parserError.py	15-Dec-20 12:20 AM	PY File	
sample_code.txt	15-Dec-20 12:40 AM	Text Document	
tiny_parser.py	15-Dec-20 11:52 AM	PY File	
tiny_scanner.py	15-Dec-20 12:39 AM	PY File	
token_list.txt	15-Dec-20 12:40 AM	Text Document	

File name: sample_code.txt

All Files (*)

Open Cancel

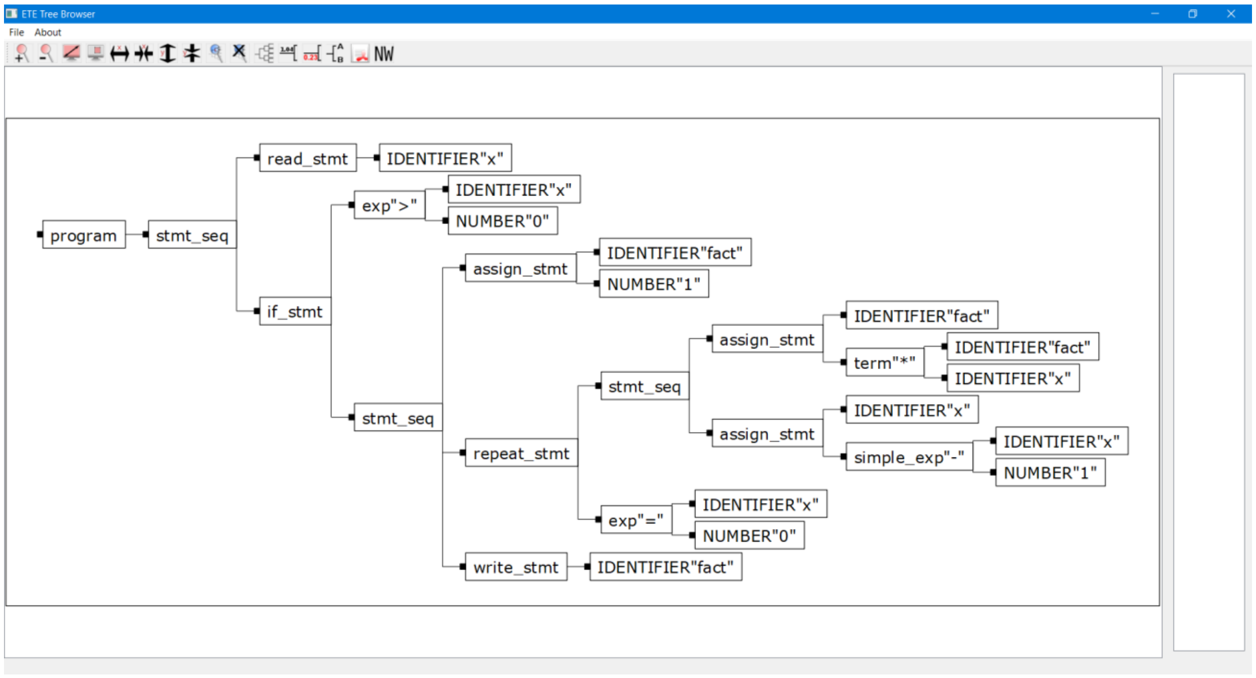
Select code from the open file window and click “Open”:



```
{Write TINY Code Here!}
write x > 5e-10
```

```
read,READ
x,IDENTIFIER
;,SEMICOLON
if,IF
x,IDENTIFIER
>,GREATERTHAN
0,NUMBER
then,THEN
fact,IDENTIFIER
:=,ASSIGN
1,NUMBER
;,SEMICOLON
repeat,REPEAT
fact,IDENTIFIER
:=,ASSIGN
fact,IDENTIFIER
*,MULT
x,IDENTIFIER
;,SEMICOLON
x,IDENTIFIER
:=,ASSIGN
x,IDENTIFIER
```

You can scroll down to see the rest of the tokens (same as the scanner example above)

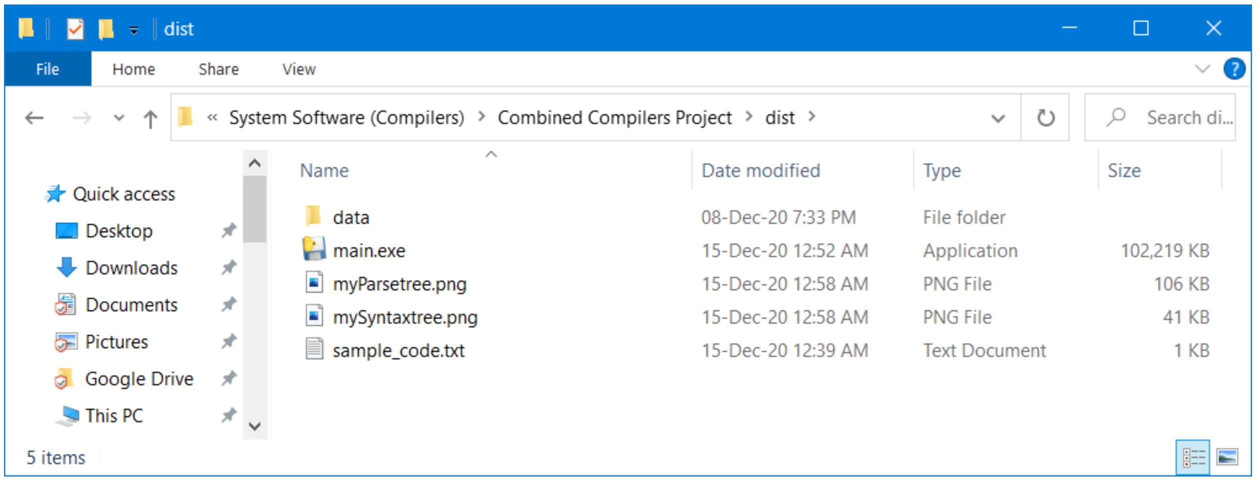


Executable:

The GUI has been compiled into a single ".exe" file.

The executable requires only the files in the "dist" folder as shown to work properly.

The contents of the "dist" folder can be moved freely anywhere and the executable would run correctly.



Name	Date modified	Type	Size
data	08-Dec-20 7:33 PM	File folder	
main.exe	15-Dec-20 12:52 AM	Application	102,219 KB
myParsetree.png	15-Dec-20 12:58 AM	PNG File	106 KB
mySyntaxtree.png	15-Dec-20 12:58 AM	PNG File	41 KB
sample_code.txt	15-Dec-20 12:39 AM	Text Document	1 KB