CS335: Milestone 1

Group 14

Members:

Name	Roll No.	Email
Supreeth Baliga	180801	supbal@iitk.ac.in
Chinmay Goyal	180206	chinmayg@iitk.ac.in
Aaryan Srivastava	180007	aaryans@iitk.ac.in
Nikhil Agarwal	180475	nikhilag@iitk.ac.in
Sanchit Agrawal	180664	sanagrwl@iitk.ac.in

Source Language (S): *C*
Implementation Language (I): *Python3*
Target Language (T): *X86 Assembly*

Repo Link: https://github.com/SupreethBaliga/Compilers/

Structure of the Repo:

```
TOP
├── src/
│   └── lexer.py
├── tests/
│   └── (Contains all tests)
├── requirements.txt
├── install.sh
├── lex.sh
├── README
└── .gitignore
```

Instructions to run:

- Install all the python modules and set up virtual environment by executing *$ install.sh*
- Activate the virtual environment by running: *$ source venvcompiler/bin/activate*
- To use the lexer execute *$ lexer.sh* and provide the test file(s) as command line arguments

Output of *'$ git show lexer '*:

```
commit 1853c619cdd21fc74f59144844cbddb259a787ac
Author: Aaryan Srivastava <aaryans@iitk.ac.in>
Date:   Sun Feb 7 20:59:53 2021 +0530

    Modified keyword list and adjusted exit codes returned in case of errors

diff --git a/lex.sh b/lex.sh
index fc4ab2c..5e16563 100755
--- a/lex.sh
+++ b/lex.sh
```

```
@@ -1,4 +1,6 @@
 #!/bin/bash
+
+STATUS=0
 if [ $# -eq 1 ] && ([ "$1" == "-h" ] || [ "$1" == "--help" ]); then
    echo "List all the programs that you want to test as $ bash lex.sh tests/test1.c tests/test2.c
    ... "
 else
@@ -6,6 +8,12 @@ else
    do
       echo $i
       python3 ./src/lexer.py $i
+      RETVAL=$?
+      if [ $RETVAL -ne 0 ];
+      then
+        STATUS=$RETVAL
+      fi;
       echo "<--------------------------------------------------------------------------->"
    done
 fi
+exit $STATUS
diff --git a/src/lexer.py b/src/lexer.py
index 81842d3..aa581b7 100644
--- a/src/lexer.py
+++ b/src/lexer.py
@@ -11,46 +11,45 @@ from tabulate import tabulate

 # add reserved keywords to this list. Expand as needed
 reserved_keywords = {
-    #basic keywords
-    'if' : 'IF',
-    'then' : 'THEN',
-    'else' : 'ELSE',
-    'for' : 'FOR',
-    'while' : 'WHILE',
-    'do' : 'DO',
-    'return' : 'RETURN',
-    'include' : 'INCLUDE',
-    'define' : 'DEFINE',
-
-    #advanced keywords
-    'switch': 'SWITCH',
-    'case': 'CASE',
-    'default' : 'DEFAULT',
-    'break' : 'BREAK',
-    'continue' : 'CONTINUE',
-    'static' : 'STATIC',
-    'auto' : 'AUTO',
-    'enum' : 'ENUM',
-    'extern' : 'EXTERN',
-    'goto' : 'GOTO',
-    'union' : 'UNION',
-
-    #data types
-    'int' : 'INT',
-    'bool' : 'BOOL',
-    'char' : 'CHAR',
-    'void' : 'VOID',
-    'struct' : 'STRUCT',
-    'double' : 'DOUBLE',
-    'float' : 'FLOAT',
-    'const' : 'CONST',
```

```
-    'long' : 'LONG'
+    'auto'      : 'AUTO',
+    'bool'      : 'BOOL',
+    'break'     : 'BREAK',
+    'case'      : 'CASE',
+    'char'      : 'CHAR',
+    'const'     : 'CONST',
+    'continue'  : 'CONTINUE',
+    'default'   : 'DEFAULT',
+    'do'        : 'DO',
+    'double'    : 'DOUBLE',
+    'else'      : 'ELSE',
+    'enum'      : 'ENUM',
+    'extern'    : 'EXTERN',
+    'float'     : 'FLOAT',
+    'for'       : 'FOR',
+    'goto'      : 'GOTO',
+    'if'        : 'IF',
+    'int'       : 'INT',
+    'long'      : 'LONG',
+    'register'  : 'REGISTER',
+    'return'    : 'RETURN',
+    'short'     : 'SHORT',
+    'signed'    : 'SIGNED',
+    'sizeof'    : 'SIZEOF',
+    'static'    : 'STATIC',
+    'struct'    : 'STRUCT',
+    'switch'    : 'SWITCH',
+    'typedef'   : 'TYPEDEF',
+    'union'     : 'UNION',
+    'unsigned'  : 'UNSIGNED',
+    'void'      : 'VOID',
+    'volatile'  : 'VOLATILE',
+    'while'     : 'WHILE'
 }

 tokens = list(reserved_keywords.values()) + [
-    'ID',       # identifier
-    # 'WS',     # denotes whitespace // may have to modify this
-                # to keep newline, space and tab separate to keep track of col no.
+    'ID',            # identifier
+    #'WS',           # denotes whitespace // may have to modify this
+                     # to keep newline, space and tab separate to keep track of col no.
     'HEXA_CONSTANT',
     'OCTAL_CONSTANT',
     'CHAR_CONSTANT',
@@ -58,7 +57,7 @@ tokens = list(reserved_keywords.values()) + [
     'INT_CONSTANT',
     'STRING_LITERAL',
     'CONSTANT',
-    'ERROR',        #to denote any kind of scanning error
+    'ERROR',         #to denote any kind of scanning error

     # Operators
     'ELLIPSIS',      # "..."
@@ -82,7 +81,7 @@ tokens = list(reserved_keywords.values()) + [
     'LE_OP',         # "<="
     'GE_OP',         # ">="
     'EQ_OP',         # "=="
-    'NE_OP'         # "!="
+    'NE_OP'          # "!="
 ]
```

```
 # Regular expression rules for simple tokens
@@ -117,6 +116,8 @@ letter = r'([a-zA-Z_])'
 hexa = r'([a-fA-F0-9])'
 exponent = r'([Ee][+-]?' + digit + r'+)'

+# Regular expression rules for complex tokens
+
 # Character Constants
 char_const = r'(\'(\\.|[^\\\']+\')'
 @TOKEN(char_const)
@@ -124,7 +125,7 @@ def t_CHAR_CONSTANT(t):
     t.type = 'CONSTANT'
     return t

-# Floating Numbers
+# Floating constants
 exponent_const = r'(' + digit + r'+' + exponent + r')'
 dec_constant = r'(' + digit + r'*[.]' + digit + r'+' + exponent + r'?)'
 float_constant = r'(' + exponent_const + r'|' + dec_constant + r')'
@@ -135,7 +136,7 @@ def t_FLOAT_CONSTANT(t):
     t.type = 'CONSTANT'
     return t

-# Hexadecimal Numbers
+# Hexadecimal Constants
 hexa_const = r'(0[xX]' + hexa + '+' + r')'
 @TOKEN(hexa_const)
 def t_HEXA_CONSTANT(t):
@@ -144,7 +145,7 @@ def t_HEXA_CONSTANT(t):
     t.type = 'CONSTANT'
     return t

-# Octal Numbers
+# Octal Constants
 octal_const = r'(0' + digit + '+' + r')'
 @TOKEN(octal_const)
 def t_OCTAL_CONSTANT(t):
@@ -153,7 +154,7 @@ def t_OCTAL_CONSTANT(t):
     t.type = 'CONSTANT'
     return t

-# Integer Numbers
+# Decimal Constants
 integer_const = r'(' + digit + '+' + r')'
 @TOKEN(integer_const)
 def t_INT_CONSTANT(t):
@@ -201,7 +202,7 @@ def t_error(t):


 ##############################################################
-# END OF TOKENS
+# END OF TOKENIZING RULES

 isError = 0
 # DRIVER CODE
@@ -226,6 +227,6 @@ for tok in lexers:

 if isError == 1:
     print(f'Errors found. Aborting scanning of {sys.argv[1]}....')
-    sys.exit()
-
```

```
-print(tabulate(table_list, headers=['Token', 'Lexeme', 'Line#', 'Column#']))
+    sys.exit(1)
+else:
+    print(tabulate(table_list, headers=['Token', 'Lexeme', 'Line#', 'Column#']))
```

Commit History:

```
1853c61 Aaryan Srivastava Sun Feb 7 20:59:53 2021 +0530
6162a04 Nikhil Agarwal Sun Feb 7 17:16:20 2021 +0530
4eabcd4 Supreeth Baliga Sun Feb 7 15:34:57 2021 +0530
b80c143 Roberticey Sun Feb 7 02:59:12 2021 +0530
803be7e Roberticey Sun Feb 7 02:37:58 2021 +0530
f3cdf5f Roberticey Sun Feb 7 02:35:15 2021 +0530
3c368a2 Chinmay Goyal Sat Feb 6 21:58:53 2021 +0530
fe9dcc5 Chinmay Goyal Sat Feb 6 21:14:09 2021 +0530
9781d00 Chinmay Goyal Sat Feb 6 20:34:50 2021 +0530
3a2c4dd Supreeth Baliga Sat Feb 6 16:48:31 2021 +0530
e2378f3 Roberticey Fri Feb 5 23:42:30 2021 +0530
c0138f2 Aaryan Srivastava Fri Feb 5 18:24:51 2021 +0530
7a2e093 Nikhil Agarwal Fri Feb 5 17:52:13 2021 +0530
2d9b1a8 Aaryan Srivastava Fri Feb 5 16:44:06 2021 +0530
5f4fbc4 Chinmay Goyal Fri Feb 5 14:32:39 2021 +0530
cad6798 Supreeth Baliga Thu Feb 4 17:27:00 2021 +0530
7926ed5 Supreeth Baliga Thu Feb 4 17:18:10 2021 +0530
dcc7522 Supreeth Baliga Thu Feb 4 16:12:38 2021 +0530
a024720 Supreeth Baliga Mon Feb 1 11:13:35 2021 +0530
8d4f0f6 Supreeth Baliga Mon Feb 1 09:04:10 2021 +0530
```

Points to Note:

- The row and column numbers are 1-indexed.
- The lexer treats *Tab* as a single column.
- The lexer has been designed to display errors if it finds stray characters.
- The lexer does not take care of macros as it assumes that the given program would be pre-processed first by gcc.