CSC 261: Principles of Programming Languages
Final Project (100 Points)
Fall 2018

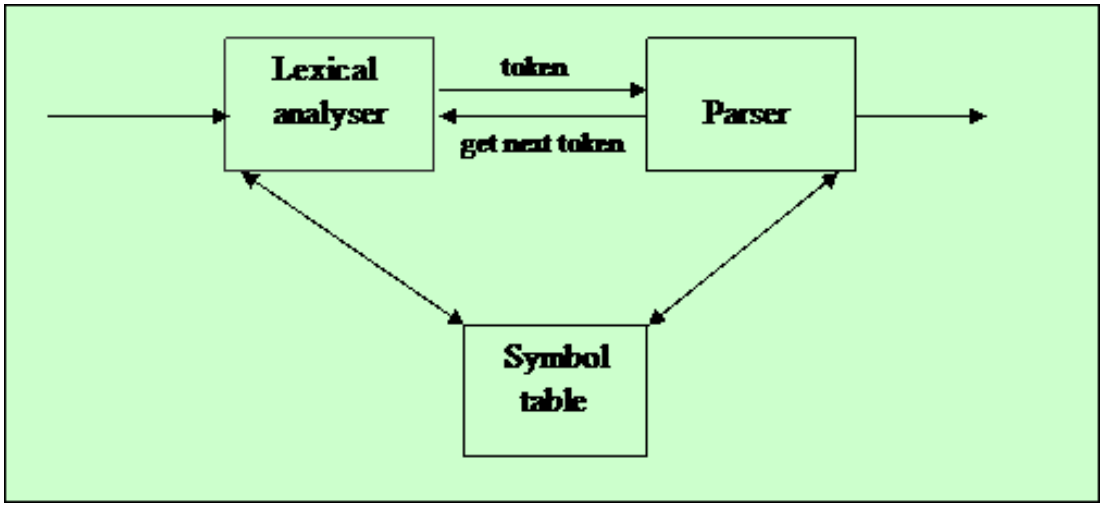
Initial Report Due Date: Monday November 19, 2018 (4p.m.)

Project Due Date: Monday December 10, 2018 (4p.m.)

Project Statement

In this final project, you are going to work in groups of 3 to implement a very simple interpreter in C for a language called **Tiny**. The language has the following features:

- There are only two data types in Tiny: *float* and *string*
- All identifiers are maintained in a symbol table (*e.g*. A, B, C, expr, etc.)



- Tiny is case-insensitive
- All expressions in Tiny are in postfix notation and are evaluates using a stack. For example,
    - (A = B C + 12.3 4 * -) evaluates to **A = (B+C) – (12.3*4)**
    - (B C ==) evaluates to (B == C)

- (B C == D 11 != &&) evaluates to **(B == C) && (D !=11)**

- Printing is done via print or println commands, each of which takes only one argument:
  - print A
  - println “This is a test”

- Comments in Tiny appear after the # character. For example:

  # this is a comment line

- Tiny has a simple postfix conditional if-statement
  - if (expr1 expr2 ==) (A 12 <) &&
    endif

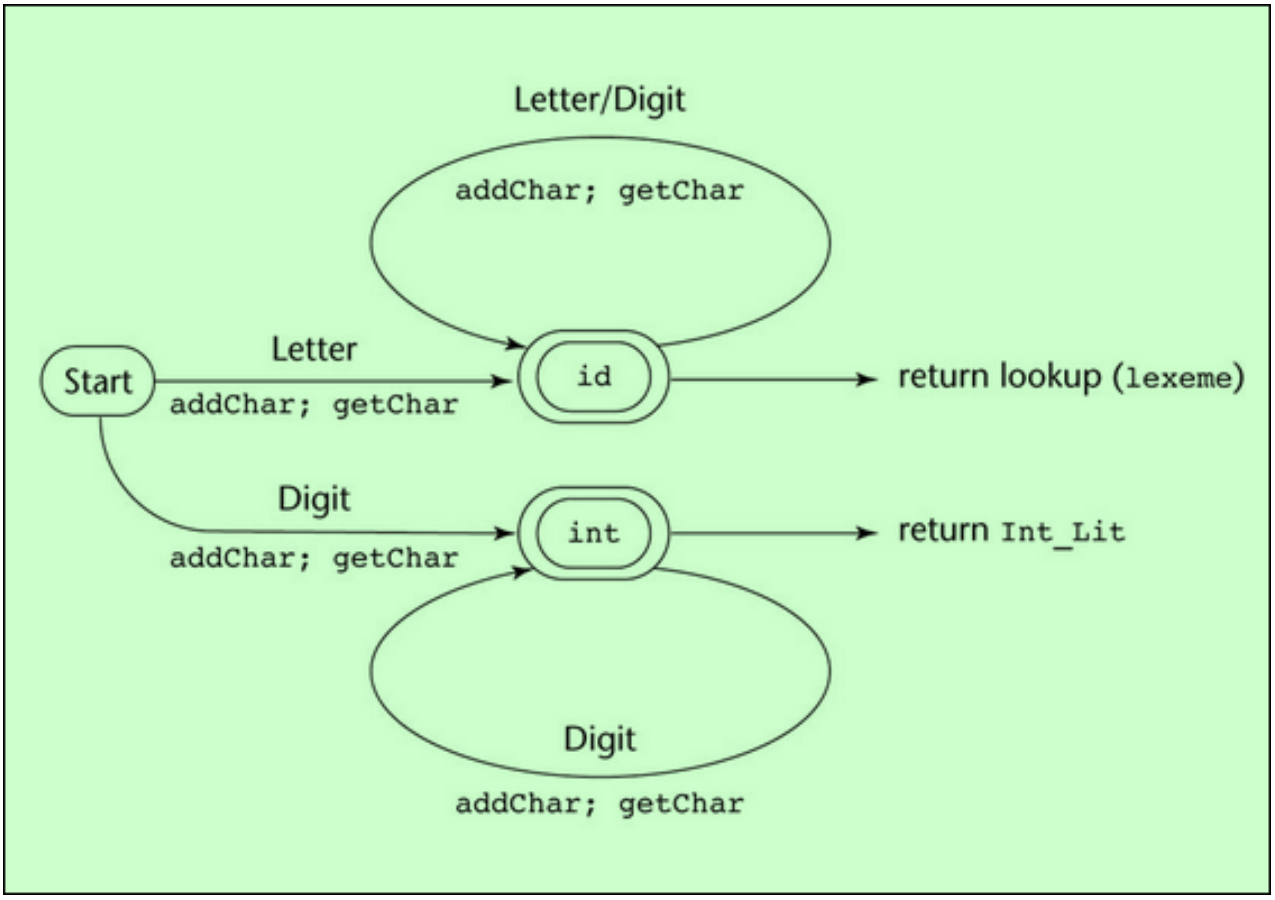
- Tiny has a simple postfix conditional while-loop
  - while (expr1 expr2 ==)
    endwhile

- Tiny does not support nesting of conditional and loop commands.

- Tiny has error-checking facilities as follows:
  - Unknown identifiers are the ones not found in the symbol table
  - Illegal or unknown commands (**legal**: while, endwhile, if, endif, print, println)
  - Illegal or unknown operands (**legal** operands: =, ==, !=, &&, ||, +, -, /, *)
  - Illegal expressions (**legal**: postfix notation)

Programming Guidelines

- You are required to use tinylex.c as the seed program in this project. The program's state diagram we have already seen in class:



Needed Programs and Data Structures

- A stack (**Stack.h**)
- A symbol table (**SymbolTable.h**)
- The interpreter **tiny.c** (should include above header files)
- A sample tiny program (**sample.tiny**)
- **Note**: When tiny.c is executed in CodeBlocks, it will automatically open **sample.tiny** and interpret it. If run from the command line, you could invoke the interpreter as: **tiny sample.tiny**

Sample Tiny Program

```
# This program sample.tiny is an example of the interpreted Tiny language



# Assignments update the symbol table
# All identifiers are global
# There are only floats and strings in the language



# The language is case-insensitive
A = 12
B = -45.6
C = "test"



# The symbol table now contains
# A         float       12
# B         float       -45.6
# C         string      "test"



# All expressions are in postfix notation
expr1 = 12 13 +
expr2 = A expr1 -



# The symbol table now contains
# A        float          12
# B        float          -45.6
# C        string       "test"
# expr1  float         25
# expr2  float        -13



# print and println take one argument
print   expr
println "Hello World"



# Error message since the identifier is not in the symbol table
print  expr3
```

```
# The language has a simple postfix conditional if statement
if (expr1 expr2 ==) (A 12 <) &&
…
endif


# The language has a simple postfix conditional while loop
while (expr1 expr2 ==)
…
endwhile
```

Grading Guidelines

Criterion	Percentage
Documentation: • Commented and documented programs • Well-written and well-organized report containing all pertinent details (**description.html**) • Is ranking statement (**ranking.html**) included?	**20%**
Error Checking	**20%**
Technical Merit: • Is the correct (intended) behavior achieved? • Are the needed data structures (e.g. the symbol table, the stack, etc.) utilized correctly and efficiently? • Is the lexical analyzer complete? • Is **Tiny's BNF or EBNF grammar** correctly implemented? • Is Tiny's **state diagram** correct?	**60%**

Submission Guidelines

1. The one-page initial report (**one per group**) should a complete list of the intended features of the language. Indicate the group leader in your proposal and whether or not you are planning to add any new features to the language in addition to what is stated above. **Any implemented extra features will be treated as extra credit.**

2. For the final report write a complete description of your project and its functionalities (**description.html**). In addition, every submitted program must be fully documented and commented with an author identification section in the beginning:

```
//*** Author's full name
//*** Course title and semester
//*** Assignment number
//*** Current date
//*** Brief description of the assignment
//*** Acknowledgment of any help received. Specifically name the sources.
```

3. In case of group projects, a statement of ranking indicating (**ranking.html**) how you rate each other's overall effort must be included. That requires that team members determine in terms of percentages how much effort was put into the project by each member of the team. Therefore, if there are two members in your team, I expect from each member the following statement (in ranking.html):

> **Member 1**: State your full name here
> My teammate and I agree that I handled **X%** of the overall project. My specific tasks included:
>
> > Task 1: I designed and implemented the program module that ... (provide exact details)
> > Task 2: I wrote ... (provide exact details)
> > Task 3: I researched ... (provide exact details)
> > Task 4: I integrated ... (provide exact details)
> > ...
>
> **Member 2**: State your full name here
> My teammate and I agree that I handled **Y%** of the overall project. My specific tasks included:
>
> > Task 1: I designed and implemented the program module that ... (provide exact details)
> > Task 2: I wrote ... (provide exact details)
> > Task 3: I researched ... (provide exact details)
> > Task 4: I integrated ... (provide exact details)

Member 3: State your full name here
My teammate and I agree that I handled **Z%** of the overall project. My specific tasks included:

Task 1: I designed and implemented the program module that ... (provide exact details)
Task 2: I wrote ... (provide exact details)
Task 3: I researched ... (provide exact details)
Task 4: I integrated ... (provide exact details)
...

4. Each team leader will email the TA the team's archived project files (ZIP or RAR). The subject line in the email should read:

   CSC 261 – Team Leader's Name – Final Project

5. In addition to above, submit hard copies of your report and project files such as ranking.html, description.html, etc., on the assigned due date **(one submission per group)**.

6. Multiple submissions are not allowed, and I also will not accept any late submissions. Inaccessible links and directories will result in the grade zero so carefully check your work before making the final submission. **Note that there are no exceptions to the rule here.**

Team Assignments

Team 1: Aydin (leader), Corbett, Diaz

Team 2: DiFilippo (leader), Grynyshin, Hartwell-Miller

Team 3: Johnson (leader), Juanillo, Kessler

Team 4: Manis (leader), Pomponio, Scuderi, Smith