CSC 3310 Concepts of Programming Languages

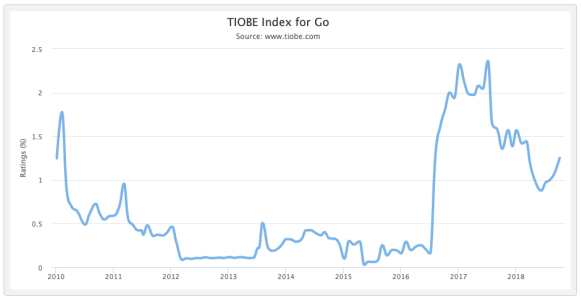
Fall 2018

Assignment Name	Go Language Programming Assignment
Due Date	November 31st, 2018
Delivery Method	Through Canvas Only
Points	10% of final grade

Motivation

Go has become a popular language, it has increased his popularity from position 20 to position 12 in the last year, according to the Tiobe index. This language was designed at Google to incorporate the "best" of available languages, avoiding some of the "worst" characteristics.

This project consists in the development of the front end of the front end of a compiler. By programming the Lexical Analyzer (Scanner) for the Hunter – Power grammar you will gain further understanding of the lexical analysis and the production of tokens needed for the Syntax Analyzer (Parser).



Taken from: https://www.tiobe.com/tiobe-index/go/

Description

Write a program in Go that takes a program written in Hunter – Power, and outputs the tokens and lexemes into a new file.

The program should run like this:

```
prompt>./go-hunter-scanner input.txt
Processing Input File input.txt
14 Tokens produced
Results in Output File input.out
prompt>
```

The tokens in the grammar are:

- STRING
- REAL_CONST
- INT_CONST
- BEGIN
- END

- WRITE
- ID
- COLON
- POINT
- ASSIGN
- TIMES
- DIVISION
- PLUS
- MINUS
- POWER
- LPAREN
- RPAREN

Given the following program written in this language:

```
$x <= "hi" :
#d <= 12 :
%r <= 3.44 :
%k <= ( #d + #d ) ^ 1.4 :
WRITE "this is the program" :
WRITE %k
```

The output file should look like:

```
ID[STRING]: x
ASSIGN
STRING: hi
COLON
ID[INT]: d
ASSIGN
INT_CONST: 12
COLON
ID[REAL]: r
ASSIGN
REAL_CONST: 3.44
COLON
ID[REAL]: k
ASSIGN
LPAREN
ID[INT]: d
PLUS
ID[INT]: d
RPAREN
POWER
REAL_CONST: 1.4
COLON
WRITE
STRING: this is the program
COLON
WRITE
ID[REAL]: k
```

Notice that the ID tokens have the partial lexeme associated (just the ID without the type symbol), along with the type (from the prefix of the ID); the constant values have their value associated too. Notice also that in the language everything is separated with a space, this is meant to make it easier to code the Scanner.

Identifiers prefixes:

- $ Represents a string type variable
- # Represents an integer type variable
- % Represents a floating-point type variable

Assignment Requirements

- Good programming practices
  - Indentation
  - Meaningful identifier naming
  - Consistent variable naming convention
- **This is an strictly individual assignment**

Delivery Method

- Submit a single ZIP named `go-hunter-power.zip` that contains the folder named `go-hunter-power` with the following files:
  - go-hunter-power.go [Scanner source code
  - test01.txt [Test Program written in Hunter – Power]
  - test02.txt [Test Program written in Hunter – Power]
- You need to use those exact same names, remember your assignment is going to check using a script, not using these names will result in a zero grade.
- Uploaded in Canvas

Assessment and Grading

Assessment will consider the following factors in the grading of the project:

- Good programming practices
- Your program will be tested with the two test programs that you will provide and some other programs that I will make, some with lexical errors, some without.
- Adherence to instructions
- Correct function of the program
- No runtime errors
- Late deliveries will have a zero mark
- Plagiarism will have a double zero mark (in addition to losing 10% of your final grade, the group that plagiarizes will lose an additional 10% of their final grade), besides there will be a report filed in the students' academic record.

Sample Run

```
local-admins-MacBook-Pro:go-hunter-scanner arias$ go build
local-admins-MacBook-Pro:go-hunter-scanner arias$ ./go-hunter-scanner test.txt
Processing Input File: test.txt
27 Tokens produced
Results in Output File: test.out
local-admins-MacBook-Pro:go-hunter-scanner arias$ cat test.out
ID[STRING]: x
ASSIGN
STRING: hi
COLON
ID[INT]: d
ASSIGN
INT_CONST: 12
COLON
ID[REAL]: r
ASSIGN
REAL_CONST: 3.44
COLON
ID[REAL]: k
ASSIGN
LPAREN
ID[INT]: d
PLUS
ID[INT]: d
RPAREN
POWER
REAL_CONST: 1.4
COLON
WRITE
STRING: this is the program
COLON
WRITE
ID[REAL]: k
```

Extra Challenge (5 bonus points)

Implement a `struct` to hold the information about the tokens:

- Lexeme
- Type
- Value
- Token

Clearly not all tokens have all four members with values. Example:

$var <= "Hello" :

This would produce:

Token 1

- `Lexeme:  var`
- `Type:  string`
- `Value:  Hello  [This you might leave empty]`
- `Token:  ID`

Token 2

- Lexeme: <=
- Type:
- Value:
- Token: ASSIGN

Token 3

- Lexeme: Hello
- Type: string
- Value: Hello
- Token: STRING

Token 4

- Lexeme: :
- Type:
- Initial Value:
- Token: COLON

You will need to add a comment on Canvas when you submit your assignment to inform me that you did the extra credit part.