SER502: Group 7

Milestone 1

Github URL: https://github.com/rajat698/SER502-Spring2022-Team7

Language Name: SKRAV (file extension: .xxx)

Team Members (Group 7) :-
Rajat Yadav
Varun Singh
Abhishek Masetty
Sahil Jambhulkar
Konark Bhad

Parsing technique: We will be implementing the parser using python3. We plan to employ the same algorithm as the DCG rules in prolog, i.e. a set of functions that consume tokens and build the tree recursively, and check validity by enforcing that all tokens are consumed.

Compiler/Interpreter runtime environment: The runtime environment will also be in python3, it will use the output of the parser and evaluate that.

Data structures used: Environment will be implemented using hash maps (dictionary) with the identifier name as the key, and the value field consisting of the identifier's value, datatype and some other information associated with it if necessary. Arrays will be used to store the list of tokens. The parse tree will be implemented using classes specific to different nodes in the tree that have links to other nodes.

Design

Declaring identifiers, assignment

```
str x4;
bool ay23 = False;
int zz111z = 2 + 3 * 5;
x = “abc”;
```

Boolean expressions

```
y = not y;
```

SER502: Group 7

Milestone 1

```
bool z = True or False;
```

Arithmetic expressions

int a = x + y / 4;

For loop

```
for-loop(i = 1; i< 10; i=i+1;) { z = z + 2; } - Traditional Way
for-loop(i in range(2,4)) { z = z+ 2; } - Using Range
```

Ternary operator

```
x == 5 ? display “lmao” : z = 2 ;
```

If-else block

```
if (x==5) { display “no else is okay” ; }
if (x==5) { display “go to else”; } else { z = 2 ; }
```

Print statements

```
display x;
display “goodbye world :(”
```

Sample program

shuru
int x;
str zzz = “Hello world”;
for-loop(i = 1; i<10; i=i+1;) {
display “something”;
}
x == 10 ? display zzz; : x = 10;
khatam

SER502: Group 7

Milestone 1

Grammar

```
# arithmetic expressions
<digit> :=  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<number> := <digit> | <digit><number>
<expr> := <number> | <id> | <expr> + <expr> | <expr> - <expr> | <expr> * <expr> |
<expr> / <expr> | <expr> % <expr> | ( <expr> )

# boolean expressions
<boolean> := True | False
<bool_expr> := <boolean> | <bool_expr> and <bool_expr> |  <bool_expr> or
<bool_expr> | not <bool_expr> | <bool_expr> == <bool_expr> | <id> <comparator>
<number> | <id> == <string> | ( <bool_expr> )
<comparator> := > | < | <= | >= | ==

# strings and identifiers
# NOTE: double quotes are not part of whitespace, they are simply used to denote the
#same
<alphabet> := a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y |
z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
<spec_char> := ! | \ | # | $ | % | & | \ | ( | ) | * | + | , | - | . | / | : | ; | < | = | > | ? | @ | ~
<whitespace> := “ “
<char> :=  <alphabet> | <digit> | <spec_char> | <whitespace>
<char_list> :=  <char> | <char><string>
<string> := “ <char_list> “ | “”

<id> := <alphabet> | <alphabet><id_suffix>
<id_suffix> := <alphabet> | <digit> | <alphabet><id_suffix> | <digit><id_suffix>

# declarations
<datatype> := int | str | bool
<stmt> := <datatype> <id> ; | <datatype> <id> = <value> ;

# assignment
<value> := <expr> | <string> | <bool_expr>
<stmt>:= <id> = <value> ;

#if-else
<stmt> := <bool_expr> ? <stmt> : <stmt>
```

SER502: Group 7

Milestone 1

```
<stmt> := if ( <bool_expr> ) { <stmt> } else { <stmt> }

#for loop
<stmt> := for-loop(<stmt> <bool_expr> ; <stmt>) { <stmt> }
<stmt> := for-loop(<id> in range(<number>,<number>)) { <stmt> }

#while loop
<stmt> := while( <bool_expr> ) {  <stmt> }

# print
<stmt> := display <id> ; | display <value> ;



# comment
<comment> := $$ <char_list> $$

# program
<prog> := shuru (<stmt> | <comment>)*  khatam
<stmt> := <stmt><stmt>
```