# JASS LANGUAGE

Team 21:

Jigisha Deven Gadhia - 1221069187

Akhila Sai Mandava – 1220311417

Sandhya Tadi - 1219346947

Venkata Naga Sonia Kalidindi - 1219398622

## OVERVIEW

- Platform & Tools
- Program Workflow
- Language Features
- Grammar
- Lexer
- Parser
- Evaluator
- Sample Runs

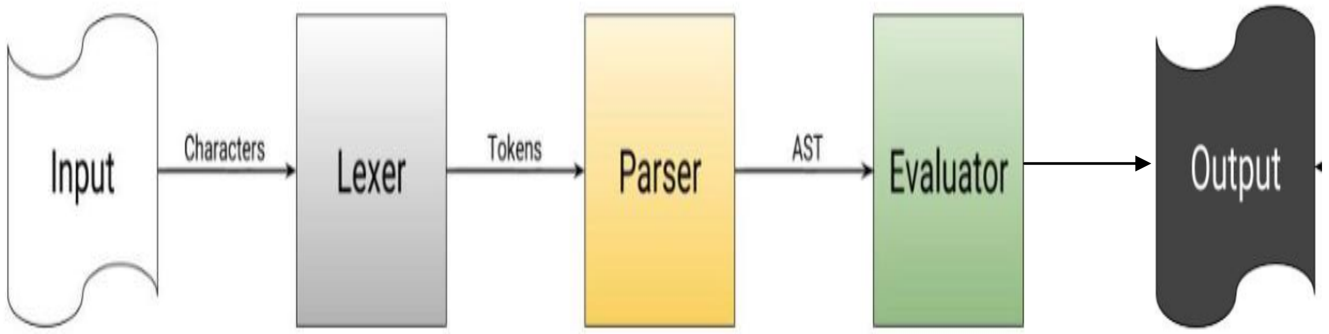
## PLATFORM AND TOOLS

| Component | Platform | Tool |
|---|---|---|
| Lexer | Python | Python IDLE |
| Parser | Prolog | Swipl |
| Evaluator | Prolog | Swipl |

## PROGRAM WORKFLOW

- Lexer takes ‘program’ as input and converts it into tokens.
- Parser takes ‘tokens’ as input and generates Abstract Syntax Tree.
- Evaluator takes ‘parse tree’ as input and produces ‘Executed Output’

## BLOCK DIAGRAM



## LANGUAGE FEATURES:

### DATATYPES

- Int – ex:- int b=6;
- String – ex:- string a=“Teamwork”
- Boolean – ex:- bool b=false;

## OPERATORS

- Unary operators (++,--)
- Arithmetic Operators(+,-,*,/,%)
- Relational operators (>>,<<,<=,>=,==)
- Logical operators (or, and, !)

## CONDITIONAL STATEMENTS

- Ternary operator

```
c = (a<<b)?{d=(b-a)}:{d=(a-b)};
```

- If and else statement

```
if(number%2 == 0){print("50 is an even number"#);}
else{print("50 is an odd number"#);}
```

## LOOPS

- For

  for (int i=0;i<< exit;i++){a=a+1;}

- While

  while ( count <= 20 )

  {print(count#);

  count=count+2;}

- For with range

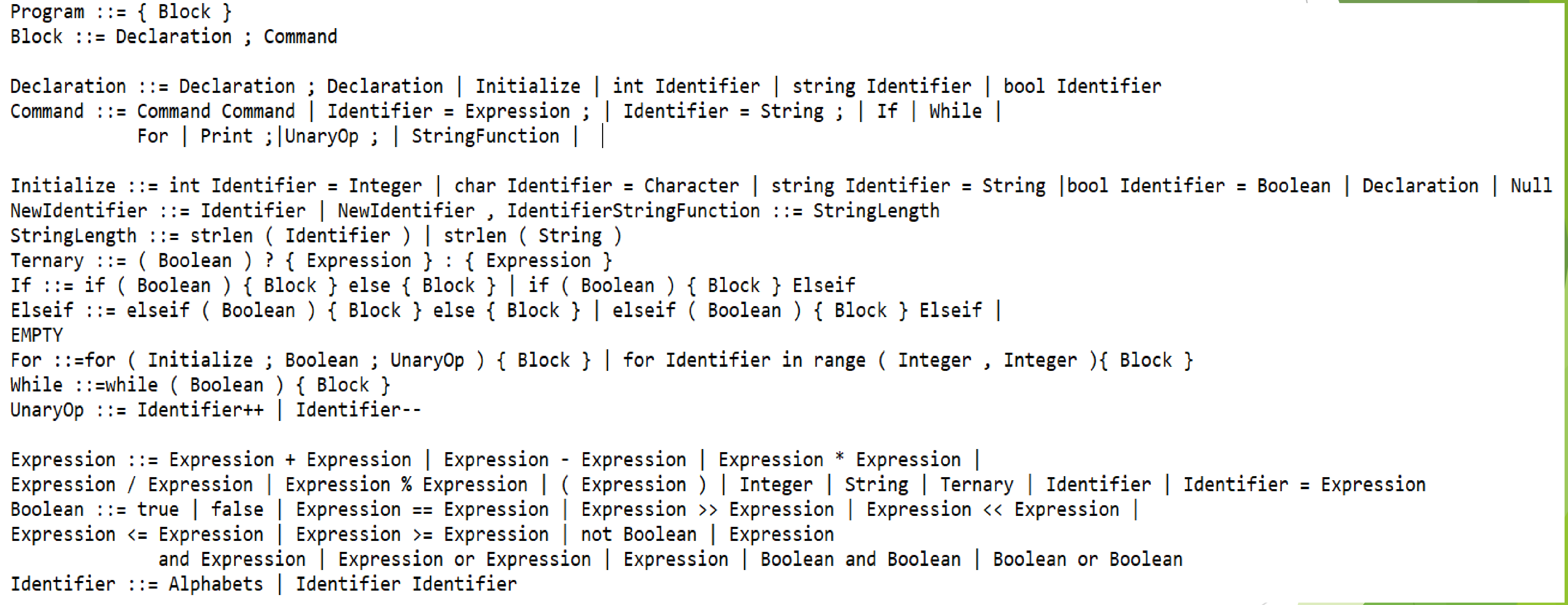
  for i in range(1,10) {num=num+1;}

## STRING FUNCTIONS

- String length:

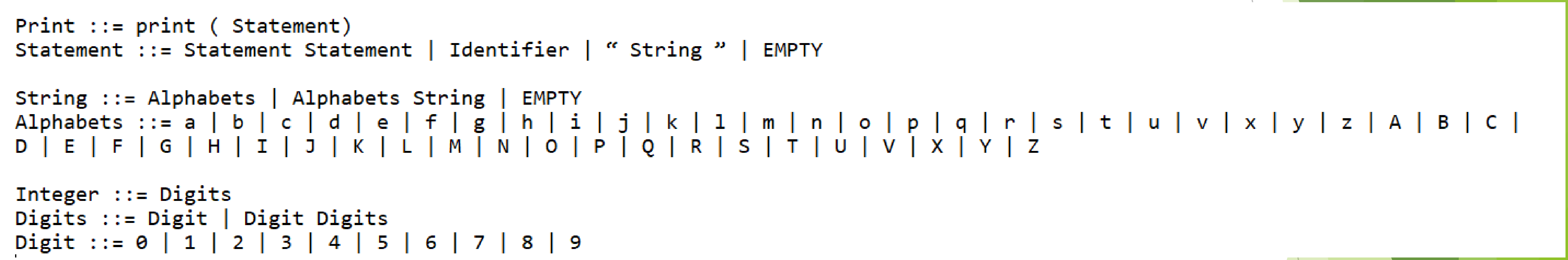
```
string str = "Hello";
int length;
print("Code for String length " #);
length = strlen(str);print("Length of Hello is: " length #);
```

# GRAMMAR



```
Program ::= { Block }
Block ::= Declaration ; Command

Declaration ::= Declaration ; Declaration | Initialize | int Identifier | string Identifier | bool Identifier
Command ::= Command Command | Identifier = Expression ; | Identifier = String ; | If | While |
        For | Print ;|UnaryOp ; | StringFunction |  |

Initialize ::= int Identifier = Integer | char Identifier = Character | string Identifier = String |bool Identifier = Boolean | Declaration | Null
NewIdentifier ::= Identifier | NewIdentifier , IdentifierStringFunction ::= StringLength
StringLength ::= strlen ( Identifier ) | strlen ( String )
Ternary ::= ( Boolean ) ? { Expression } : { Expression }
If ::= if ( Boolean ) { Block } else { Block } | if ( Boolean ) { Block } Elseif
Elseif ::= elseif ( Boolean ) { Block } else { Block } | elseif ( Boolean ) { Block } Elseif |
EMPTY
For ::=for ( Initialize ; Boolean ; UnaryOp ) { Block } | for Identifier in range ( Integer , Integer ){ Block }
While ::=while ( Boolean ) { Block }
UnaryOp ::= Identifier++ | Identifier--

Expression ::= Expression + Expression | Expression - Expression | Expression * Expression |
Expression / Expression | Expression % Expression | ( Expression ) | Integer | String | Ternary | Identifier | Identifier = Expression
Boolean ::= true | false | Expression == Expression | Expression >> Expression | Expression << Expression |
Expression <= Expression | Expression >= Expression | not Boolean | Expression
           and Expression | Expression or Expression | Expression | Boolean and Boolean | Boolean or Boolean
Identifier ::= Alphabets | Identifier Identifier
```

# GRAMMAR

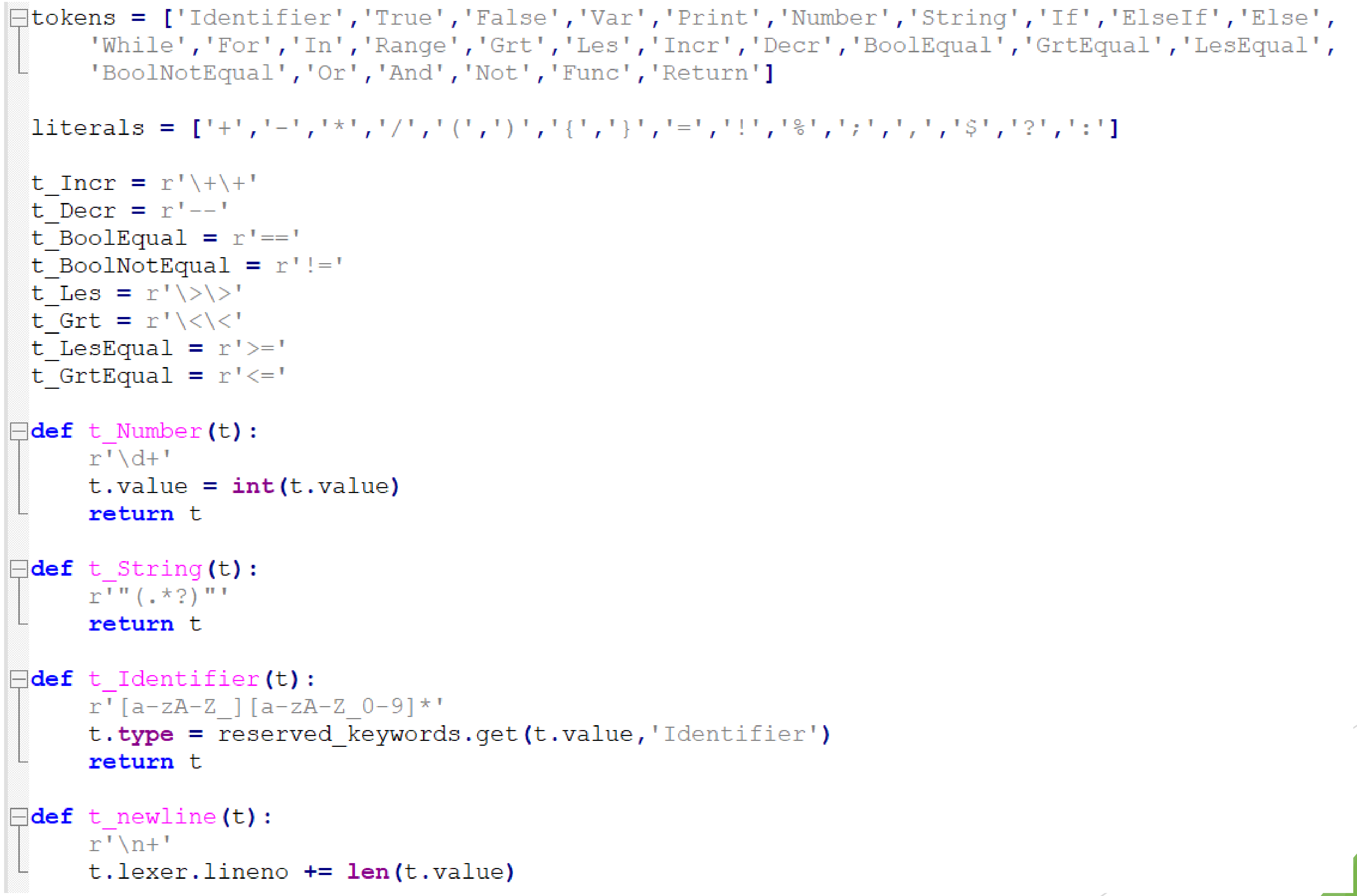


```
Print ::= print ( Statement)
Statement ::= Statement Statement | Identifier | “ String ” | EMPTY

String ::= Alphabets | Alphabets String | EMPTY
Alphabets ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | 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 | X | Y | Z

Integer ::= Digits
Digits ::= Digit | Digit Digits
Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

## LEXER

- Removes extra spaces.
- Convert programs to tokens.

# GLIMPSE OF LEXER CODE

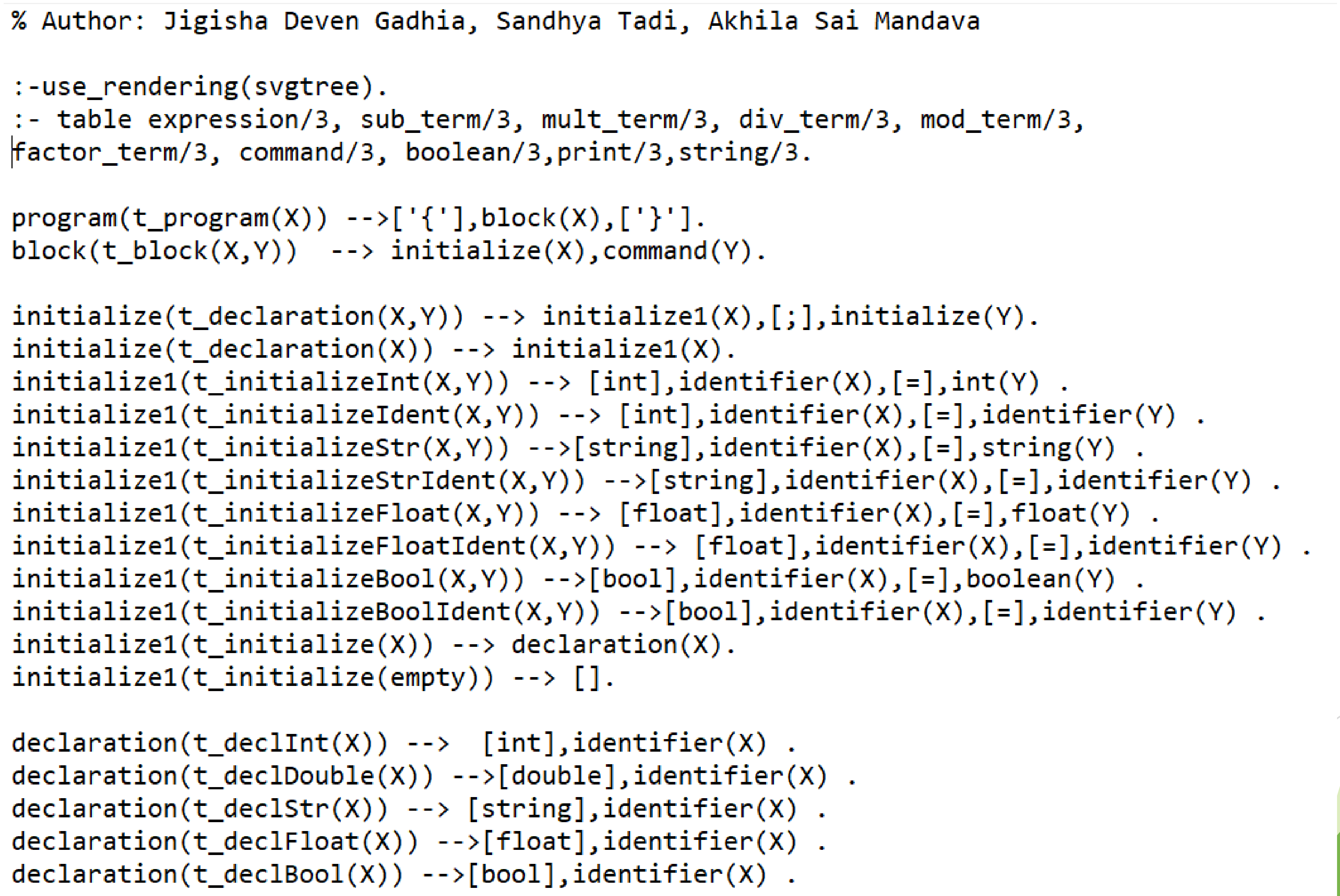


```
tokens = ['Identifier','True','False','Var','Print','Number','String','If','ElseIf','Else',
    'While','For','In','Range','Grt','Les','Incr','Decr','BoolEqual','GrtEqual','LesEqual',
    'BoolNotEqual','Or','And','Not','Func','Return']

literals = ['+','-','*','/','(',')','{','}','=','!','%',';',',','$','?',':']

t_Incr = r'\+\+'
t_Decr = r'--'
t_BoolEqual = r'=='
t_BoolNotEqual = r'!='
t_Les = r'\>\>'
t_Grt = r'\<\<'
t_LesEqual = r'>='
t_GrtEqual = r'<='

def t_Number(t):
    r'\d+'
    t.value = int(t.value)
    return t

def t_String(t):
    r'"(.*?)"'
    return t

def t_Identifier(t):
    r'[a-zA-Z_][a-zA-Z_0-9]*'
    t.type = reserved_keywords.get(t.value,'Identifier')
    return t

def t_newline(t):
    r'\n+'
    t.lexer.lineno += len(t.value)
```

## PARSER

- Input – Tokens List
- Output – Abstract Syntax Tree
- Top-Down Approach

# GLIMPSE OF PARSER CODE

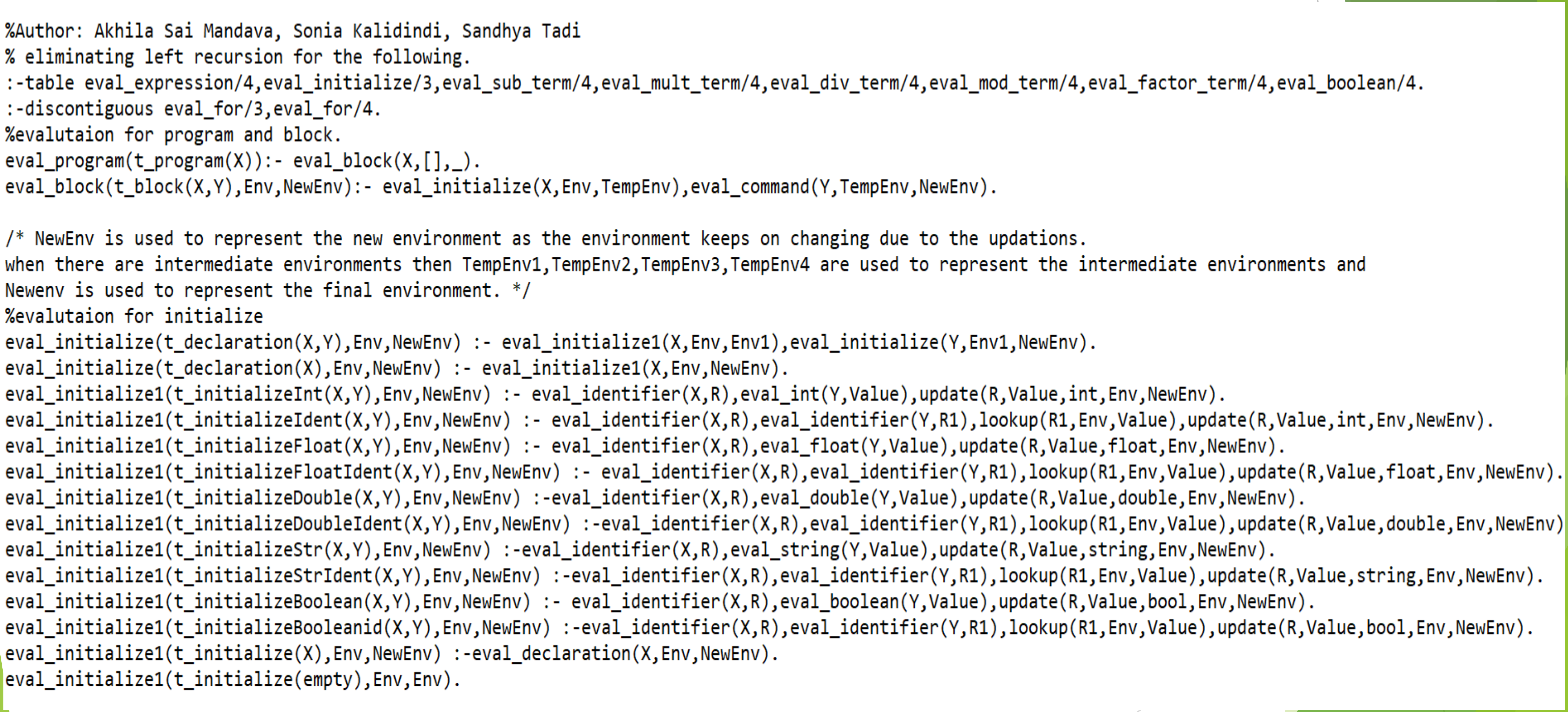


```
% Author: Jigisha Deven Gadhia, Sandhya Tadi, Akhila Sai Mandava

:-use_rendering(svgtree).
:- table expression/3, sub_term/3, mult_term/3, div_term/3, mod_term/3,
factor_term/3, command/3, boolean/3,print/3,string/3.

program(t_program(X)) -->['{'],block(X),['}'].
block(t_block(X,Y))  --> initialize(X),command(Y).

initialize(t_declaration(X,Y)) --> initialize1(X),[;],initialize(Y).
initialize(t_declaration(X)) --> initialize1(X).
initialize1(t_initializeInt(X,Y)) --> [int],identifier(X),[=],int(Y) .
initialize1(t_initializeIdent(X,Y)) --> [int],identifier(X),[=],identifier(Y) .
initialize1(t_initializeStr(X,Y)) -->[string],identifier(X),[=],string(Y) .
initialize1(t_initializeStrIdent(X,Y)) -->[string],identifier(X),[=],identifier(Y) .
initialize1(t_initializeFloat(X,Y)) --> [float],identifier(X),[=],float(Y) .
initialize1(t_initializeFloatIdent(X,Y)) --> [float],identifier(X),[=],identifier(Y) .
initialize1(t_initializeBool(X,Y)) -->[bool],identifier(X),[=],boolean(Y) .
initialize1(t_initializeBoolIdent(X,Y)) -->[bool],identifier(X),[=],identifier(Y) .
initialize1(t_initialize(X)) --> declaration(X).
initialize1(t_initialize(empty)) --> [].

declaration(t_declInt(X)) -->  [int],identifier(X) .
declaration(t_declDouble(X)) -->[double],identifier(X) .
declaration(t_declStr(X)) --> [string],identifier(X) .
declaration(t_declFloat(X)) -->[float],identifier(X) .
declaration(t_declBool(X)) -->[bool],identifier(X) .
```

## EVALUATOR

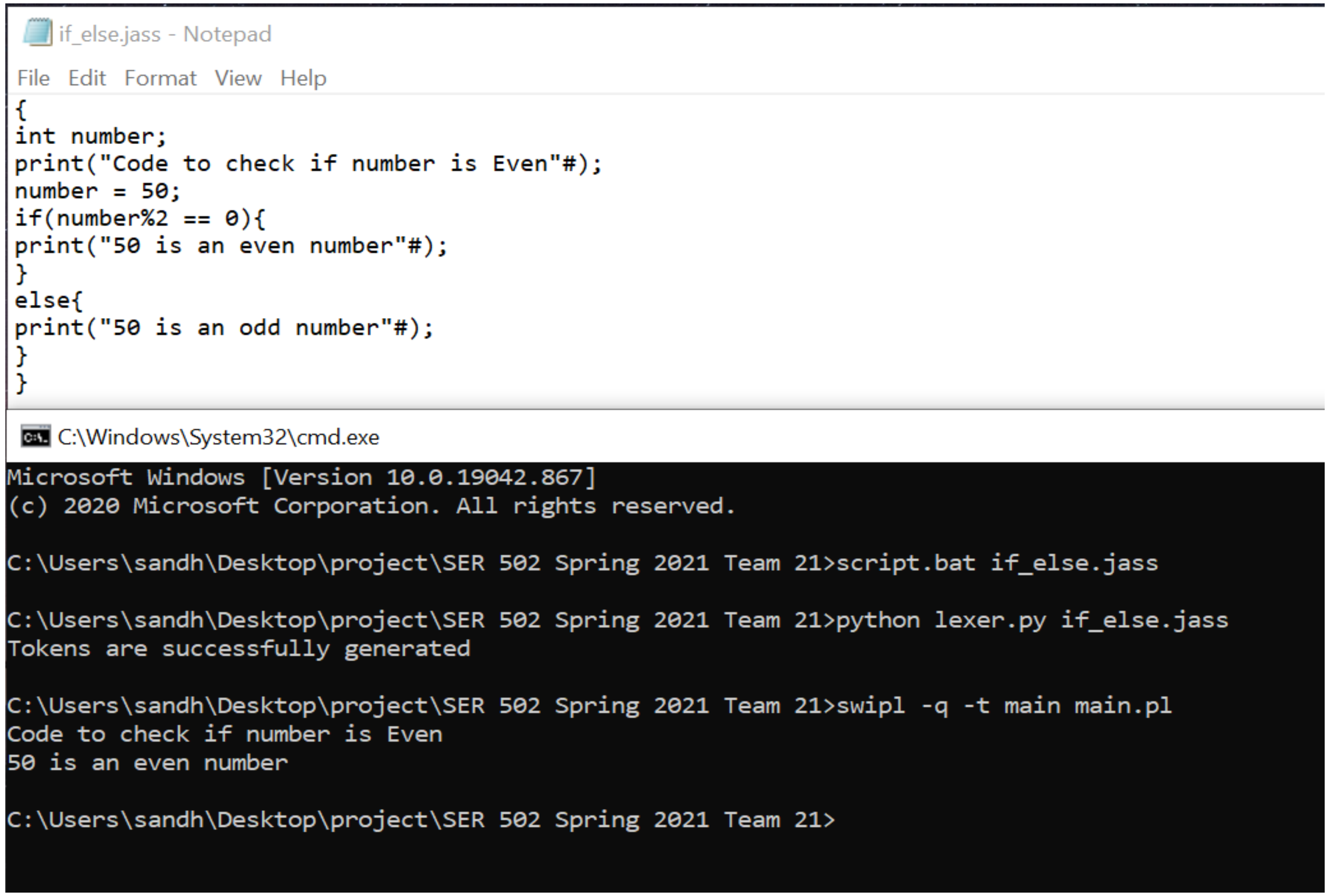
- Input – Abstract Syntax Tree
- Output – Executed Output

# GLIMPSE OF EVALUATOR CODE



```
%Author: Akhila Sai Mandava, Sonia Kalidindi, Sandhya Tadi
% eliminating left recursion for the following.
:-table eval_expression/4,eval_initialize/3,eval_sub_term/4,eval_mult_term/4,eval_div_term/4,eval_mod_term/4,eval_factor_term/4,eval_boolean/4.
:-discontiguous eval_for/3,eval_for/4.
%evalutaion for program and block.
eval_program(t_program(X)):- eval_block(X,[],_).
eval_block(t_block(X,Y),Env,NewEnv):- eval_initialize(X,Env,TempEnv),eval_command(Y,TempEnv,NewEnv).

/* NewEnv is used to represent the new environment as the environment keeps on changing due to the updations.
when there are intermediate environments then TempEnv1,TempEnv2,TempEnv3,TempEnv4 are used to represent the intermediate environments and
Newenv is used to represent the final environment. */
%evalutaion for initialize
eval_initialize(t_declaration(X,Y),Env,NewEnv) :- eval_initialize1(X,Env,Env1),eval_initialize(Y,Env1,NewEnv).
eval_initialize(t_declaration(X),Env,NewEnv) :- eval_initialize1(X,Env,NewEnv).
eval_initialize1(t_initializeInt(X,Y),Env,NewEnv) :- eval_identifier(X,R),eval_int(Y,Value),update(R,Value,int,Env,NewEnv).
eval_initialize1(t_initializeIdent(X,Y),Env,NewEnv) :- eval_identifier(X,R),eval_identifier(Y,R1),lookup(R1,Env,Value),update(R,Value,int,Env,NewEnv).
eval_initialize1(t_initializeFloat(X,Y),Env,NewEnv) :- eval_identifier(X,R),eval_float(Y,Value),update(R,Value,float,Env,NewEnv).
eval_initialize1(t_initializeFloatIdent(X,Y),Env,NewEnv) :- eval_identifier(X,R),eval_identifier(Y,R1),lookup(R1,Env,Value),update(R,Value,float,Env,NewEnv).
eval_initialize1(t_initializeDouble(X,Y),Env,NewEnv) :-eval_identifier(X,R),eval_double(Y,Value),update(R,Value,double,Env,NewEnv).
eval_initialize1(t_initializeDoubleIdent(X,Y),Env,NewEnv) :-eval_identifier(X,R),eval_identifier(Y,R1),lookup(R1,Env,Value),update(R,Value,double,Env,NewEnv)
eval_initialize1(t_initializeStr(X,Y),Env,NewEnv) :-eval_identifier(X,R),eval_string(Y,Value),update(R,Value,string,Env,NewEnv).
eval_initialize1(t_initializeStrIdent(X,Y),Env,NewEnv) :-eval_identifier(X,R),eval_identifier(Y,R1),lookup(R1,Env,Value),update(R,Value,string,Env,NewEnv).
eval_initialize1(t_initializeBoolean(X,Y),Env,NewEnv) :- eval_identifier(X,R),eval_boolean(Y,Value),update(R,Value,bool,Env,NewEnv).
eval_initialize1(t_initializeBooleanid(X,Y),Env,NewEnv) :-eval_identifier(X,R),eval_identifier(Y,R1),lookup(R1,Env,Value),update(R,Value,bool,Env,NewEnv).
eval_initialize1(t_initialize(X),Env,NewEnv) :-eval_declaration(X,Env,NewEnv).
eval_initialize1(t_initialize(empty),Env,Env).
```

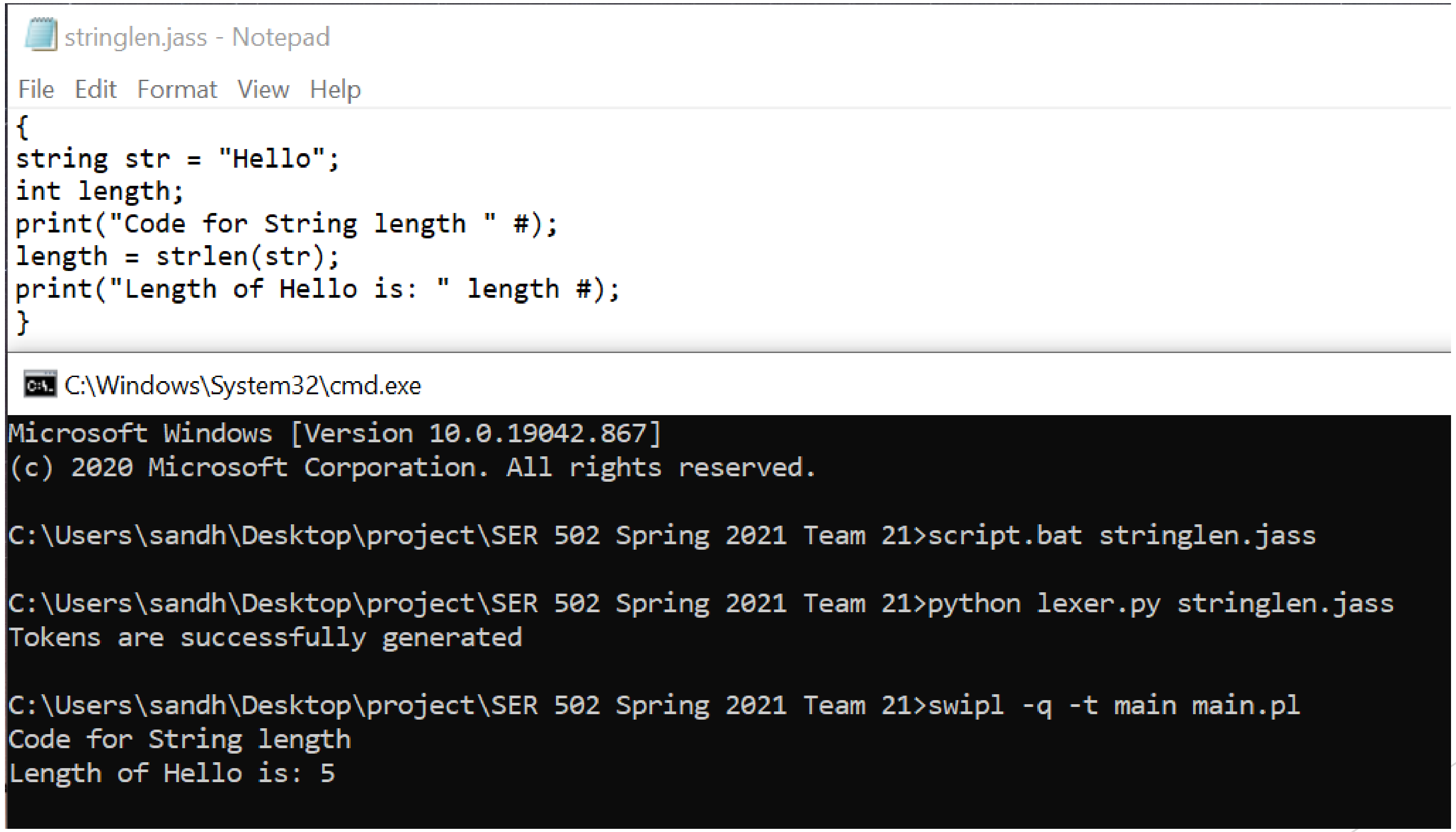
## SAMPLE RUN:- 1



```
{
int number;
print("Code to check if number is Even"#);
number = 50;
if(number%2 == 0){
print("50 is an even number"#);
}
else{
print("50 is an odd number"#);
}
}
```

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>script.bat if_else.jass

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>python lexer.py if_else.jass
Tokens are successfully generated

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>swipl -q -t main main.pl
Code to check if number is Even
50 is an even number

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>
```

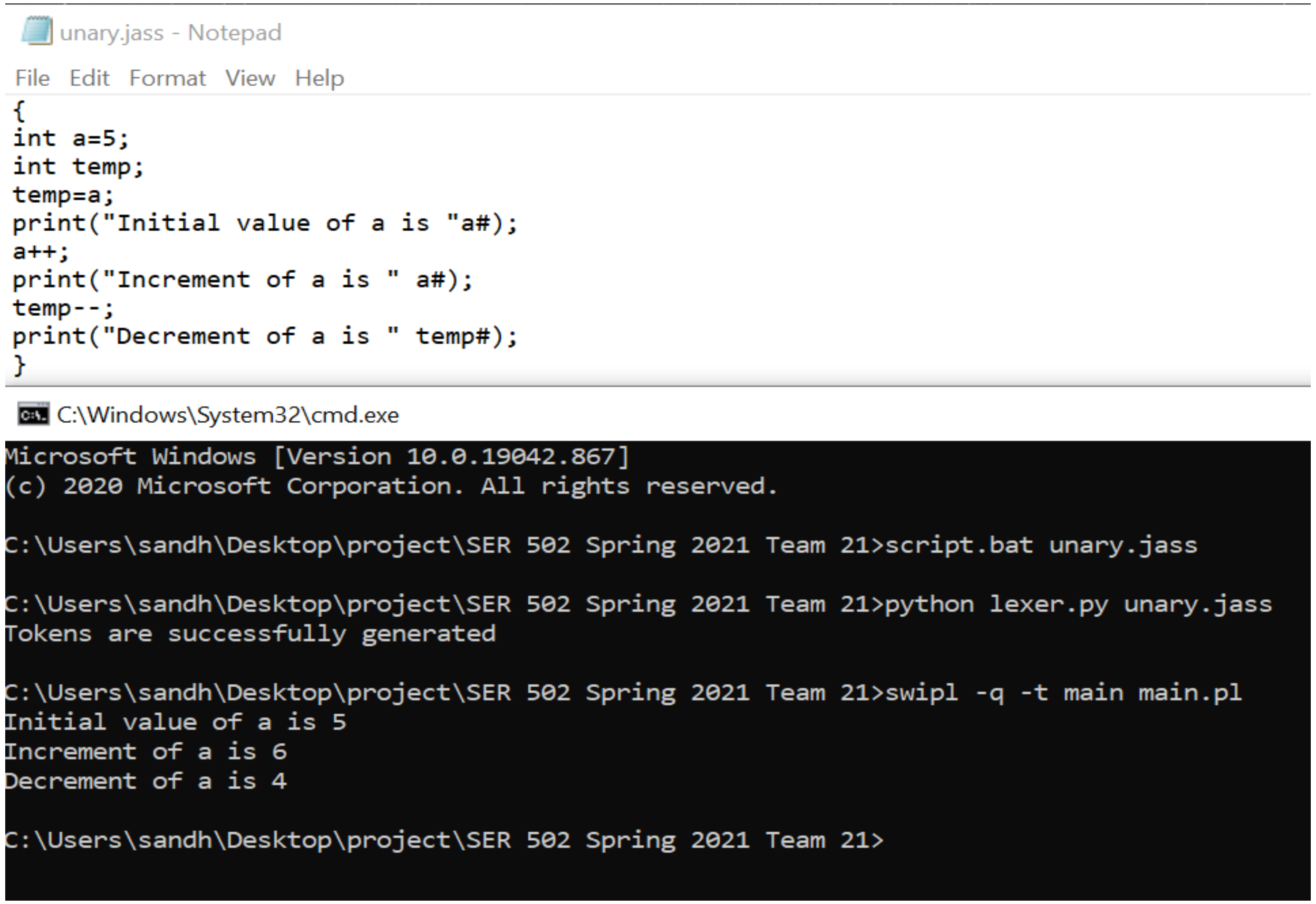
## SAMPLE RUN:- 2



```
{
string str = "Hello";
int length;
print("Code for String length " #);
length = strlen(str);
print("Length of Hello is: " length #);
}
```

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>script.bat stringlen.jass

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>python lexer.py stringlen.jass
Tokens are successfully generated

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>swipl -q -t main main.pl
Code for String length
Length of Hello is: 5
```

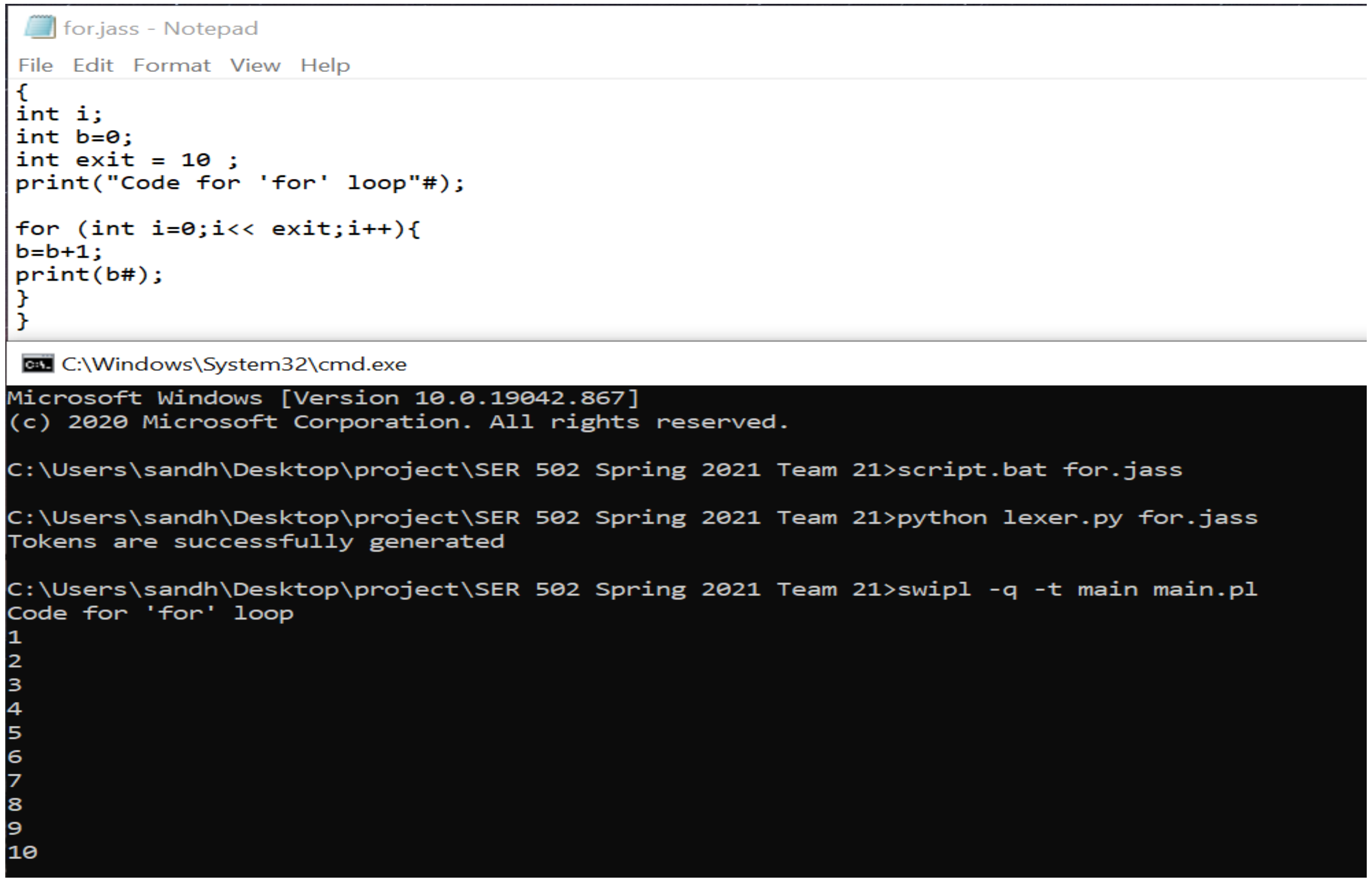
## SAMPLE RUN:- 3



```
{
int a=5;
int temp;
temp=a;
print("Initial value of a is "a#);
a++;
print("Increment of a is " a#);
temp--;
print("Decrement of a is " temp#);
}
```

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>script.bat unary.jass

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>python lexer.py unary.jass
Tokens are successfully generated

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>swipl -q -t main main.pl
Initial value of a is 5
Increment of a is 6
Decrement of a is 4

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>
```

## SAMPLE RUN:- 4



for.jass - Notepad

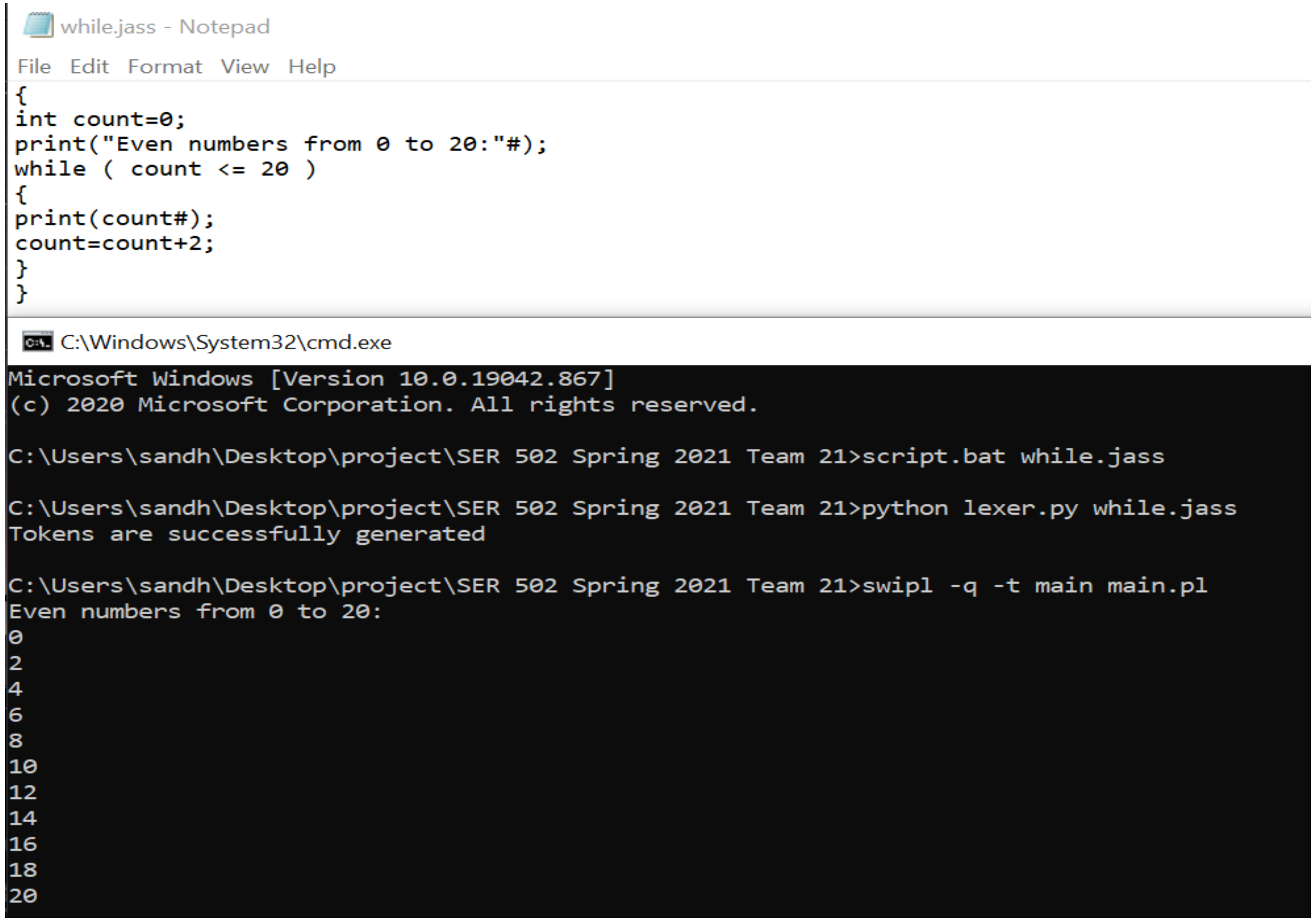
File Edit Format View Help

```
{
int i;
int b=0;
int exit = 10 ;
print("Code for 'for' loop"#);

for (int i=0;i<< exit;i++){
b=b+1;
print(b#);
}
}
```

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>script.bat for.jass

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>python lexer.py for.jass
Tokens are successfully generated

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>swipl -q -t main main.pl
Code for 'for' loop
1
2
3
4
5
6
7
8
9
10
```

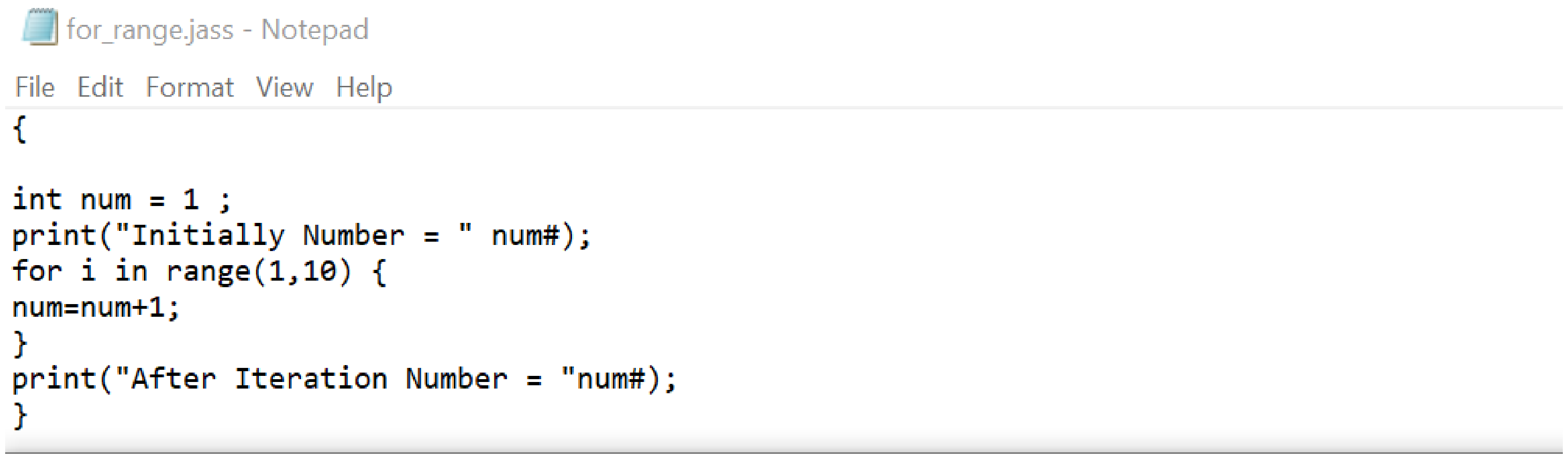
## SAMPLE RUN:- 5



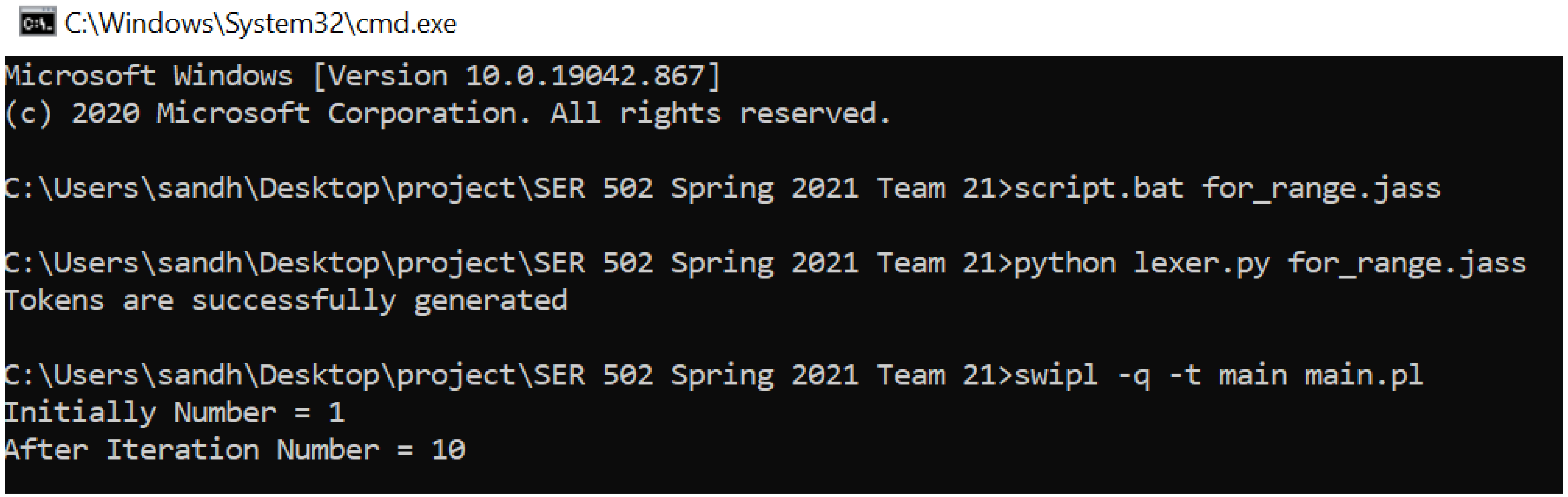
```
{
int count=0;
print("Even numbers from 0 to 20:"#);
while ( count <= 20 )
{
print(count#);
count=count+2;
}
}
```

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>script.bat while.jass

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>python lexer.py while.jass
Tokens are successfully generated

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>swipl -q -t main main.pl
Even numbers from 0 to 20:
0
2
4
6
8
10
12
14
16
18
20
```

## SAMPLE RUN:- 6



```
{

int num = 1 ;
print("Initially Number = " num#);
for i in range(1,10) {
num=num+1;
}
print("After Iteration Number = "num#);
}
```



```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>script.bat for_range.jass

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>python lexer.py for_range.jass
Tokens are successfully generated

C:\Users\sandh\Desktop\project\SER 502 Spring 2021 Team 21>swipl -q -t main main.pl
Initially Number = 1
After Iteration Number = 10
```

THANK YOU