

HABESOME

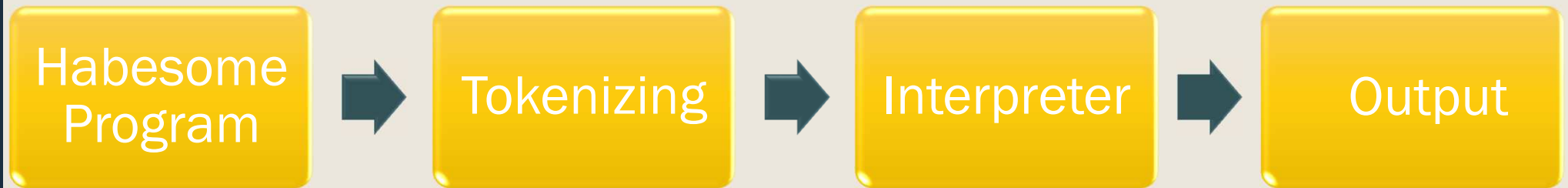
SER 502: Team 18

- 1) Melissa Day
- 2) Sowmya Madabhushi
- 3) Harshita Kajal
- 4) Behnaz Sabbaghi

Overview:

- Habesome Introduction
- Grammar
- Lexical Analysis
- Parser
- Interpreter
- Sample Programs

Habesome Design Flow



Grammar Description:

- Tokens in grammar are broken down into 1) terminals 2) non terminals
- The grammar handles precedence.
- Rules are defined for the grammar are:

Grammar

```
1  P is Program
2  K is Block
3  D is Declaration
4  DT is data types
5  SL is Statement List
6  S is Statement
7  A is Assignment Statement
8  IF is If Statement
9  W is While Statement
10 B is Boolean Expression
11 E is Arithmetic Expression
12 EX is Multiplication or Division Arithmetic Expression
13 I is Identifier
14 N is Number
15 INT is integer data type
16 BOOL is boolean type
17 DG is Digit
18 L is Letter
19 RES is resolution of addition or subtraction
20 RESMD is resolution of multiplication or division
21 FACTOR is a part (factor) of a multiplication or division expression
22
23 P ::= 'start' K 'end'
24 K ::= '[' D SL ']' | // Removed extra period
25       '[' SL ']'
26 D ::= DT I '.' D |           // Fixed for adding assignment
27       DT I '.' |
28       DT A '.' D |
29       DT A '.'
30
31 DT ::= 'int' | 'bool' // fixed/updated
```

```
32
33 SL ::= S '.' SL |
34       S '.' |
35       K
36
37 S ::= A | IF | W
38
39 A ::= I 'is' E
40
41 IF ::= 'if' '(' B ')' K 'else' K
42
43 W ::= 'while' '(' B ')' K
44
45 B ::= 'true' |
46       'false' |
47       E '=' E |
48       '!!' E |
49       E '<<' E |
50       E '>>' E
51
52 =====
53 // Left associativity fixed
54 E ::= EX RES |
55       EX
56
57 RES ::= '+' EX RES |
58       '-' EX RES |
59       '+' EX | '-' EX
60
61 // Left associativity fixed
62 EX ::= FACTOR RESMD |
63       FACTOR
```

```

52  =====
53  // Left associativity fixed
54  E ::= EX RES |
55      EX
56
57  RES ::= '+' EX RES |
58      '-' EX RES |
59      '^' EX | '^' EX
60
61  // Left associativity fixed
62  EX ::= FACTOR RESND |
63      FACTOR
64
65  RESND ::= '*' FACTOR RESND |
66      '/' FACTOR RESND |
67      '**' FACTOR | '/' FACTOR
68
69  FACTOR ::= I | N
70
71  =====
72
73  I ::= L I |
74      L
75
76  N ::= DG N |
77      DG
78
79  DG ::= [0-9]+
80
81  L ::= ('a'..'z' | 'A'..'Z')*
```

```

3  program : block ;
4
5  block : '[' ( declaration '.' statementList ) ']' |
6      '[' statementList ']';
7  declaration : datatype statementList '.' declaration |
8      datatype statementList '.';
9
10 datatype: INT | BOOLEAN;
11
12 statementList : statement '.' statementList |
13     statement '.' |
14     block;
15
16 statement : assignment | ifStatement | whileloop;
17
18 assignment : identifier 'is' expression;
19
20 ifStatement : 'if' '(' bool ')' block 'else' block;
21
22 whileloop : 'while' '(' bool ')' block;
23
24 bool : 'True' |
25     'False' |
26     expression '=' expression |
27     '!' expression |
28     expression '<<' expression |
29     expression '>>' expression;
30
31

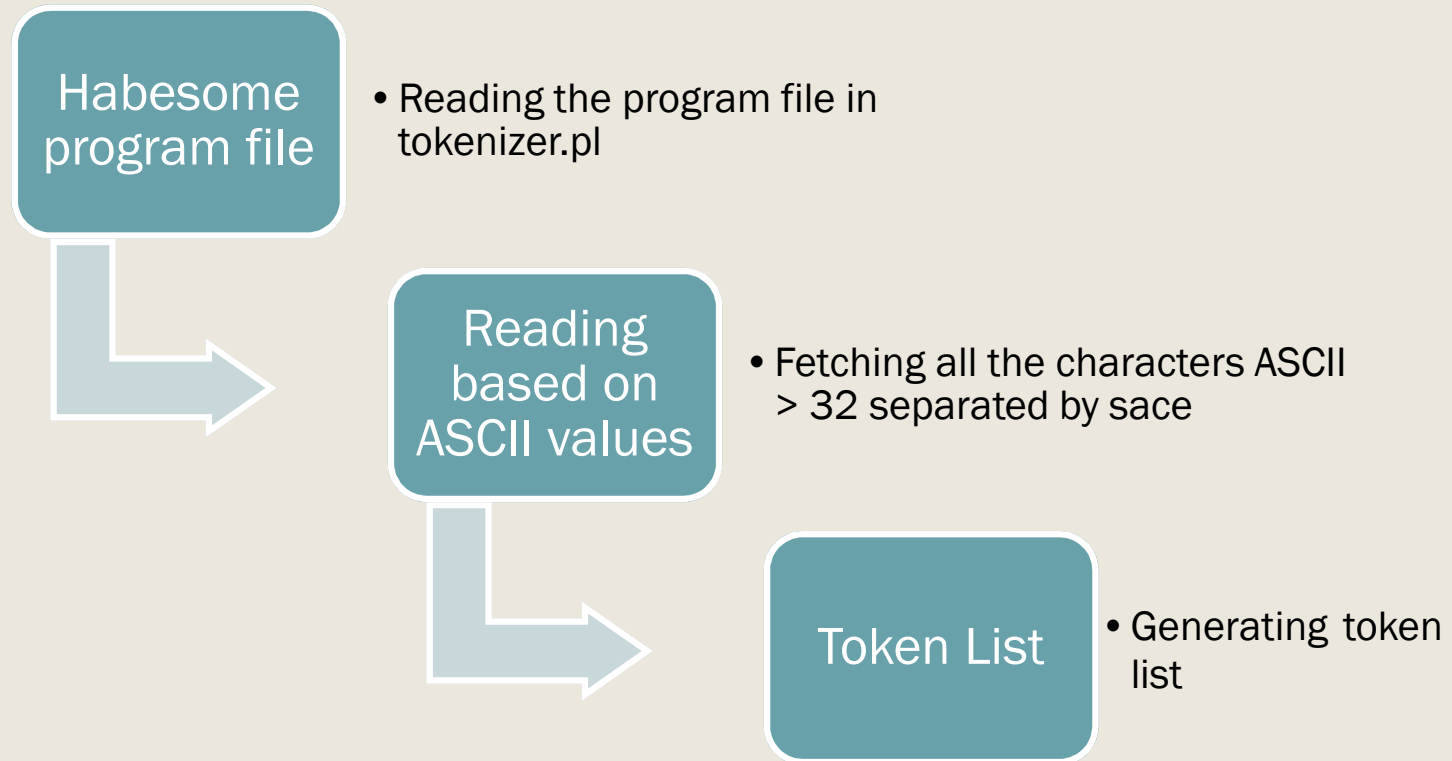
```

```

31 expression : term '+' expression |
32
33 term '-' expression |
34 term;
35
36
37 term : identifier '*' term |
38     number '*' term |
39     identifier '/' term |
40     number '/' term |
41     identifier |
42     number ;
43
44 identifier: L identifier |
45     L ;
46
47 number : DG number |
48     DG ;
49 INT: [0-9]+ ;
50 DG : [0-9]+ ;
51 BOOLEAN: 'True' | 'False';
52
53 L : ('a'..'z' | 'A'..'Z')*;
54
55

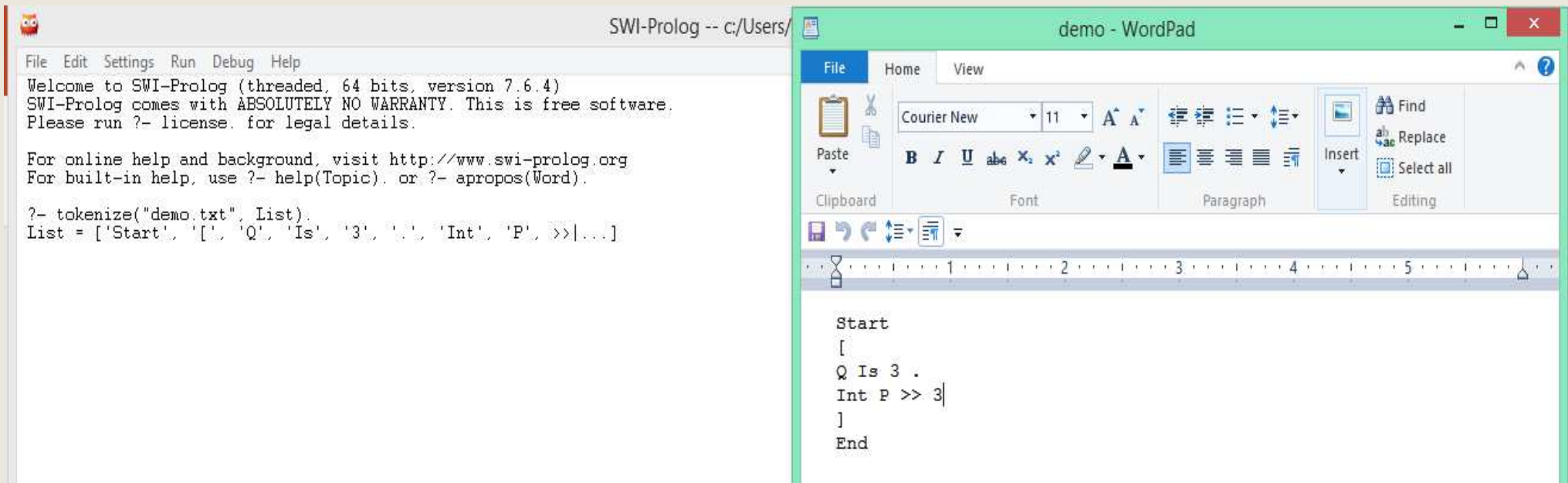
```

Lexical Analysis



Sample output for tokenizer

- All the words in the file are separated by space
- Tokens generated are separated by comma



The image shows two windows side-by-side. The left window is titled 'SWI-Prolog -- c:/Users/' and contains the following text:

```
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- tokenize("demo.txt", List).
List = ['Start', '[', 'Q', 'Is', '3', '.', 'Int', 'P', '>>|...]
```

The right window is titled 'demo - WordPad' and shows the text from the file 'demo.txt' as it appears in a text editor:

```
Start
[
Q Is 3 .
Int P >> 3|
]
End
```

Parser

- Parser takes in the input generated from tokenizer
- Parse tree generated using a top-down approach
- The rules for parser are written in DCG (Definite Clause Grammar)

```
start
[
int p is 2.
q is 3.
]
end

% Execution -
?- L = ['start', '[', 'int', 'p', 'is', '2', '.', 'q', 'is', '3', '.', ']', 'end'], program(P, L, []).

% o/p -
L = [start, '[', int, p, (is), '2', ('.'), q, (is), '3', ('.'), ']', end],
P = parsetree(blockdec(deca(dtype1(int), assign(iden(letter(p))), arithexp(exp(factor(nu(digit('2'))))))),
slist(stmtassign(assign(iden(letter(q))), arithexp(exp(factor(nu(digit('3'))))))))
% -
```

Interpreter

- Receives the parse tree as input from the parser
- Uses this predicate to run the interpreter

```
% Interpreter program predicate
interpreter(PTokens, FinalEnv, OutputFileName) :-
    program(PTree, PTokens, []),
    writeOutputToNewFile(OutputFileName, "Parse Tree: "),
    writeOutputToExistingFile(OutputFileName, PTree),
    Env = [],
    evalProgram(PTree, Env, FinalEnv),
    writeOutputToExistingFile(OutputFileName, "Final Environment: "),
    writeOutputToExistingFile(OutputFileName, FinalEnv).
```

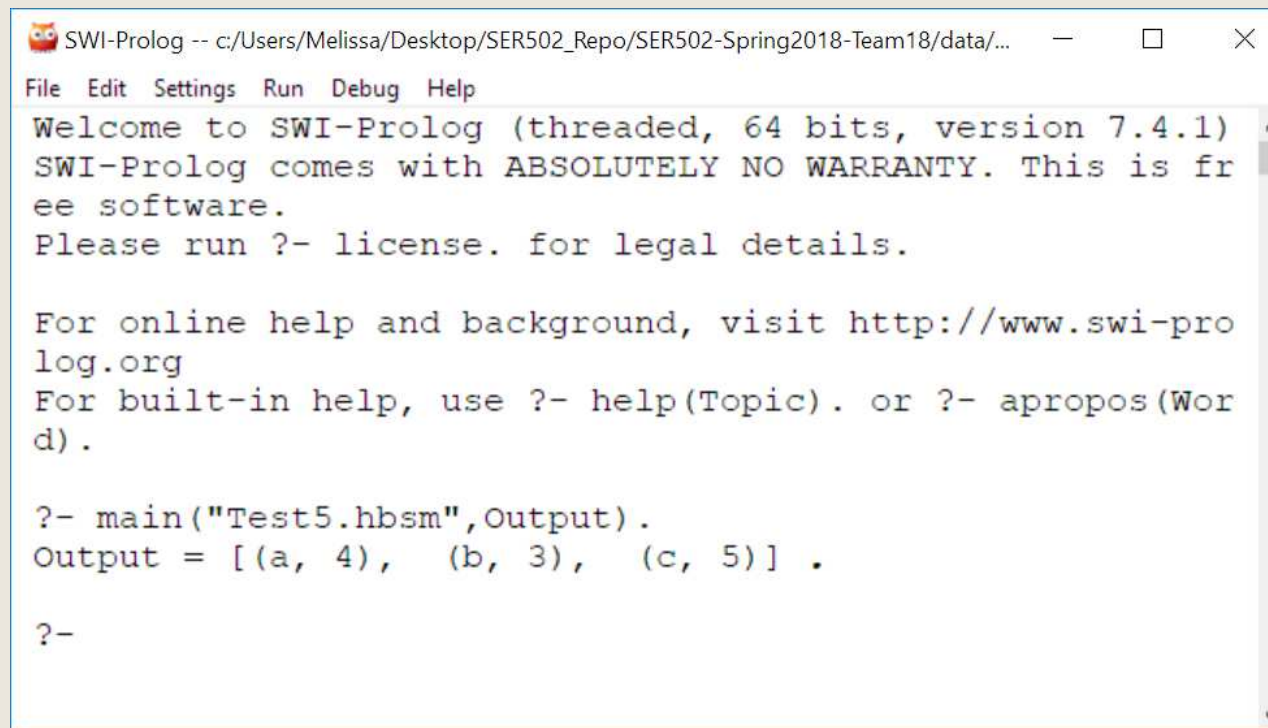
- Initializes the environment to an empty list
- Adds the new environment as pairs to the list

Writing a Habesome Program:

- Statements end with the period '.' (fullstop)
- All the words in the program must be written with one character space (including the period and special characters like +, (,), etc ..)
- Assignment should be done for the same datatype
- Datatypes for a variable cannot be changed later in the program

Steps to run:

- The program should be saved as a .hbsm file
- In SWI-prolog we write the following execution query:



```
SWI-Prolog -- c:/Users/Melissa/Desktop/SER502_Repo/SER502-Spring2018-Team18/data/...
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 7.4.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is fr
ee software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-pro
log.org
For built-in help, use ?- help(Topic). or ?- apropos(Wor
d) .

?- main("Test5.hbsm",Output).
Output = [(a, 4), (b, 3), (c, 5)] .

?-
```

Steps to run:

- Program upon successful execution produces output in a .txt file format called “ProgramOutput.txt”
- File contains parse tree and output containing the final program environment

Steps to run:

- It generates an output file in the same directory with name “Program Output” which displays the required output and also the parse tree

```
% Runs the entire program from tokenizing to output.  
% Saves results of parsetree and final environment to an output program called  
% "ProgramOutput.txt"  
% Expected Parameters:  
% InputFileName: Filename where the program input is stored, Extension should be .hbsm  
  
main(InputFileName, FinalEnvironment) :- tokenize(InputFileName, Tokens),  
                                         interpreter(Tokens, FinalEnvironment, "ProgramOutput.txt").
```

Sample Program

```
start
[
if ( true )
[
d is 6 * 7 + 2 * 8 .
q is 3 .
].
]
end
```

A screenshot of a SWI-Prolog window. The title bar reads "SWI-Prolog -- c:/Users/Melissa/Desktop/SER502_Repo/SER502-Spring2018-Team18/data/...". The menu bar includes "File", "Edit", "Settings", "Run", "Debug", and "Help". The main text area displays a welcome message for SWI-Prolog version 7.4.1, stating it comes with absolutely no warranty and is free software. It prompts the user to run "?- license." for legal details and provides links for online help (http://www.swi-prolog.org) and built-in help (?- help(Topic) or ?- apropos(Word)). Below this, it shows the execution of "?- main('Test4.hbsm', Output).", resulting in "Output = [(d, 58), (q, 3)]". The prompt "?- " is followed by a black square cursor.

```
SWI-Prolog -- c:/Users/Melissa/Desktop/SER502_Repo/SER502-Spring2018-Team18/data/...
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 7.4.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is fr
ee software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-pro
log.org
For built-in help, use ?- help(Topic). or ?- apropos(Wor
d).

?- main("Test4.hbsm",Output).
Output = [(d, 58), (q, 3)] .

?- 
```


Sample Program

Parse Tree:

```
parsetree(block(slist(stmtif(ifstate(boolexptrue(true),block(slistrec(stmtassign(assign(iden(letter(d)),  
arithexpl(expl(factor(num(digit(6))),resmul2(factor(num(digit(7))))),resarithsum(expl(factor(num(digit(2))),  
resmul2(factor(num(digit(8)))))))))slist(stmtassign  
(assign(iden(letter(q)),arithexp(exp(factor(num(digit(3))))))))))))))
```

Final Environment:

```
[(d,58),(q,3)]
```