Ahsanullah University of Science and Technology
Department of CSE
Fall 2024
Course Name: Formal Language and Compilers Lab Course No: CSE 4130
Assignment 4

Syntax errors are very common in source programs. The main purpose of this session is to write programs to detect and report simple syntax errors.

Suppose, we have a C source program with various syntax errors. Let's try to analyze the different types of error it contains. (You can also try to compile the following program to see what types of errors are reported by the compiler):

```
/* A program fragment*/

float x1 =  3.125;;;
double x1;

/* Definition of function f1 */
double  f1(float a, int int x)

{if(x<<x1)
double z;;
else z = 0.01;}}

else return z;

}
/* Beginning of 'main' */
int main(void)
{{{{
int n1; double z;
n1=25; z=f1(n1);
printf("Hello");

y = n1 + 5;
for(;;){
}

}
```

=> x1 is declared twice globally

=> int is declared twice

=> else without a previous if

=> Mismatched parenthesis

=> y is not declared

=> Mismatched parenthesis

Assignment:

Assume that a given C source program has already been scanned, filtered, lexically analyzed, and tokenized, as was done in earlier sessions. In addition, line numbers have been assigned to the source code lines for generating proper error messages.

We now aim to detect common syntax errors such as:

1. Duplication of tokens except parentheses or braces or semicolons,
2. Unbalanced braces or parentheses problem
3. Unmatched ***'else'*** problem (else without a previous if)
4. Duplicate identifier declarations.
5. Undeclared identifier declarations.

6. Checking ***for*** loop constructs (e.g., double ; in for(;;)).
   Those who can fully check the correctness of *for* loop syntax will get bonus marks.

Sample Input:
A C source program with several syntax errors.

```
1.  /* A program fragment*/
2.  float x1 =  3.125;;;
3.  double x1;
4.  /* Definition of function f1 */
5.  double  f1(float a, int int x)
6.  {if(x<<x1)
7.  double z;;
8.  else z = 0.01;}}
9.  else return z;
10. }
11. /* Beginning of 'main' */
12. int main(void)
13. {{{{
14. int n1; double z;
15. n1=25; z=f1(n1);
16. printf("Hello");
17. y = 2 + 5;
18. for(;;){
19. }
20. }
21.
```

Sample Output:
As you may have already figured out, Assignment 4 requires all the previous assignments to be completed first before detecting syntax errors. The expected output should demonstrate the following four steps in order:

Step 1: Scanned and filtered source program (output of Assignment 1).
Step 2: Separated and tokenized lexemes (output of Assignment 2).
Step 3: Generated Symbol Table (output of Assignment 3).
Step 4 (Main task of Assignment 4): Detect and report syntax errors with their respective line numbers. Errors must be listed in sorted order by line number.

Ahsanullah University of Science and Technology
Department of CSE
Fall 2024
Course Name: Formal Language and Compilers Lab Course No: CSE 4130
Assignment 4

Guidelines:

1. You need to modify assignment 1 and 2 a little bit to incorporate the line numbers in your codes. You can tokenize the line numbers with a dedicated token so that your compiler doesn't confuse them with identifiers or numeric constants.
2. For every ***'else'*** there must be an ***'if'*** that occurs earlier. You can easily detect it by maintaining a count of ***'if'*** and ***'else'***. Using a Stack can be really helpful in this case. You need to perform a little bit more modification to handle the ***'else if'*** case.
3. In a similar way, you can keep track of unbalanced parentheses/braces.
4. Both undeclared and duplicated identifiers can easily be detected by the Symbol Table.