Session 4: Detecting Simple Syntax Errors

I. OBJECTIVES:

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.

II. DEMONSTRATION OF USEFUL RESOURCES:

Sample programs related to detection of simple syntax errors will be demonstrated.

III. LAB EXERCISE:

Write programs to detect the following syntax errors.

1. Duplicate subsequent keywords, keeping in mind exceptions like 'else if'.
2. Unbalanced curly braces.

IV. ASSIGNMENT #4:

Suppose, a given C source program has been scanned, filtered, lexically analyzed and tokenized as that were done in earlier sessions. In addition, line numbers have been assigned to the source code lines for generating proper error messages. As the first step to Syntax Analysis, we now perform detection of simple syntax errors like duplication of tokens, except parentheses or braces, unbalanced braces or parentheses problem, unmatched 'else' problem, etc. Duplicate identifier declarations must also be detected with the help of the Symbol Table.

Sample Input: Sample code segment with numerous syntax errors,

```
/* A program fragment*/

float     x1 = 3.125;;;
/* Definition of function f1 */
double  f1(float a, int int x)
{if(x<x1)
double z;;
else z =          0.01+x*5.5;}}
else   return z;
}
/* Beginning of 'main'      */
int main(void)
{{{{
int n1;  double z;
n1=25;  z=f1(n1);}
```

Intermediate Output: Recognized tokens in the lines of code.

```
1
2
3 float id x1 = 3.125 ; ; ;
4
5 double id f1 ( float id a , int int id x )
6 { if ( id x < id x1 )
7 double id z ; ;
8 else id z = 0.01 + id x * 5.5 ; } }
9 else return id z ;
10 }
11
12 int id main ( void )
13 { { { {
14 int id n1 ; double id z ;
15 id n1 = 25 ; id z = id f1 ( id n1 ) ; }
16
```

Sample Output: Types of detected errors

```
Duplicate token at line 3, Misplaced '}' at line 8, Unmatched 'else' at line 9, etc.
```

<u>Guidelines:</u>

1. Unbalanced braces or parentheses problem in an arithmetic or relational expression can be detected during tokenization in a simple way by counting the openings and closings:
2. Unmatched ‘else’ problem in its simplest form may also be detected by counting ‘if’s and ‘else’s: For every ‘else’ there must be an ‘if’ that occurs earlier.
3. Undeclared identifiers and duplicate identifier declarations in the same scope can be detected during Symbol Table construction.