Bangladesh Agricultural University

Department of Bioinformatics Engineering

Course Code & Title: CSM 3222 Compiler Lab

Level-3, Semester-2, July-December/2024

Deadline: 03 Jan, 2026 11:59 PM — Lab 2 Tasks — Platform: Google Classroom

For each task, write a Flex (Lex) program to perform the required lexical analysis. Use regular expressions and actions inside the .l file. The program should read input from an external file (input.txt).

1. Write a Flex program that reads a source file and counts the total number of:

   - Characters
   - Words
   - Lines

   The scanner must correctly handle spaces, tabs, and newline characters.

   Sample Input:

   ```
   int main() {
       printf("Hello World\n");
   }
   ```

   Sample Output:

   ```
   Characters: 43
   Words: 6
   Lines: 3
   ```

2. Given an input line of C code, write a Flex program to identify and print all C keywords present in the line. Non-keyword identifiers must not be printed.

   Sample Input:

   ```
   int main() { float num = 2.5;
     return 0; }
   ```

   Sample Output:

   ```
   int: Keyword
   float: Keyword
   return: Keyword
   ```

3. Write a Flex program to extract all identifiers from a given line and classify them as valid or invalid. First, write the regular expression for a valid identifier. Then implement the rule using Flex.

   Sample Input:

   ```
   Enter a line:
   id1, _id2, 3id, id 5
   ```

   Sample Output:

   ```
   id1: Valid Identifier
   _id2: Valid Identifier
   3id: Invalid Identifier
   id: Valid Identifier
   5: Invalid Identifier
   ```

4. Write a Flex program to detect and classify numeric constants into:

   - Integer constants
   - Floating-point constants

   Use appropriate regular expressions.

Sample Input:

```
Enter a line:
a = 10; b = 2.75; c = 300;
```

Sample Output:

```
10: Integer
2.75: Float
300: Integer
```

5. Given an input line, write a Flex program to identify and print the following operators:

   Operators Include:

   - **Assignment:** =
   - **Arithmetic:** +, -, *, /, %
   - **Relational:** ==, !=, >, <, >=, <=
   - **Logical:** &&, ||, !

   Sample Input:

   ```
   Enter a line:
   if (a >= 10 && b != 5) a = a + 1;
   ```

   Sample Output:

   ```
   >= : Relational Operator
   && : Logical Operator
   != : Relational Operator
   = : Assignment Operator
   + : Arithmetic Operator
   ```

6. Write a Flex program to identify and print all punctuation/special symbols from the following set:

   { } ( ) ; , [ ] .

   Sample Input:

   ```
   Enter a line:
   if(a%2==0) printf("%d is even", a
      );
   ```

   Sample Output:

   ```
   ( : Special Symbol
   ) : Special Symbol
   ( : Special Symbol
   , : Special Symbol
   ) : Special Symbol
   ; : Special Symbol
   ```

7. Write a Flex program to detect and print:

   - Single-line comments (// ...)
   - Multi-line comments (/* ... */)

   The program must also detect unterminated comments and report them as lexical errors.

   Sample Input:

   ```
   Enter a line or comment:
   /* multi line comment */
   ```

   Sample Output:

   ```
   Multi-line Comment
   ```

8. Write a Flex program to detect valid and invalid string literals and character constants. Also detect errors such as:

   - Unterminated strings ”hello
   - Empty char constants ”
   - Multiple characters ’ab’

Sample Input:

```
"a"
"hi"
"hello
'a'
''
'ab'
'a
```

Sample Output:

```
"a" : Valid String Literal
"hi" : Valid String Literal
" : Unterminated String Literal
'a' : Valid Character Constant
'' : Empty Character Constant (Invalid)
'ab' : Multiple Characters (Invalid)
' : Unterminated Character Constant (Invalid)
```

9. Write a Flex-based lexical analyzer that scans a file and classifies each lexeme into:

   - Keyword
   - Identifier
   - Operator
   - Constant
   - Special Symbol
   - String Literal
   - Comment

   Generate a token stream in the format:

   ```
   <token_type, lexeme>
   ```

   Also count how many tokens of each category appear.

   Sample Input:

   ```
   int x = 10;
   float y = 2.5;
   printf("Hello");
   ```

   Sample Output:

   ```
   <int, keyword>
   <x, identifier>
   <=, operator>
   <10, constant>
   <;, special_symbol>

   <float, keyword>
   <y, identifier>
   <=, operator>
   <2.5, constant>
   <;, special_symbol>

   <printf, identifier>
   <("Hello"), string_literal>
   <;, special_symbol>

   Token Counts:
   Keywords: 2
   Identifiers: 3
   Constants: 2
   Operators: 2
   Special symbols: 3
   String literals: 1
   Invalid tokens: 0
   ```

10. Write a Flex program that detects and reports common lexical errors:

    - **Invalid identifiers:** 2sum, a-bc, @total

- **Unterminated strings:** "hello
- **Unclosed comments:** /* comment
- **Invalid characters:** @, $, # (in identifiers)

You must define valid patterns using regular expressions.

Sample Input:

```
int 2sum = 10;
char *s = "hello;
float x@ = 2.5;
```

Sample Output:

```
Error: Invalid identifier '2sum'
Error: Unterminated string
    literal "hello
Error: Invalid character '@' in
    identifier x@
```