As part of the first phase of your project you have to implement the syntax analyser and parser for a the decaf programming language.
Your analyser should successfully parse all valid programs, and give an error if it encounters an invalid program.
Output (to stdout) “Success” on a successful parse, and “Syntax Error” in case of an error.

For more info about Decaf : Decaf Manual

Create a new file “flex_output.txt” which on encountering a symbol outputs the respective Output value on a new line. Note the following table is not exhaustive. You must generate similar output for **all tokens** as specified in the Decaf manual.

flex_output.txt

Symbol Found	Output
boolean	BOOLEAN_DECLARATION
callout	CALLOUT
class	CLASS
false	BOOLEAN: false
int	INT_DECLARATION
true	BOOLEAN: true
<id>	ID: <id>
<int_literal>	INT: <int_literal>
<char_literal>	CHARACTER: <char_literal>
<string_literal>	STRING: <string_literal>

Priority Order (Highest to Lowest, the ones on same level have same priority):

1. Unary Minus
2. !
3. * , / , %
4. + -
5. > <

All operators are “left”-associative.

Similarly, create a new file “bison_output.txt” which on encountering a rule, outputs the respective output value on a new line. Again, note the following table is not exhaustive. You must generate similar output for **all rules** as specified in the Decaf manual.

bison_output.txt

<program>	PROGRAM ENCOUNTERED
int <id>	INT DECLARATION ENCOUNTERED. ID=<id>

int <id> [<size>]	INT DECLARATION ENCOUNTERED.
	ID=<id>
	SIZE=<size>

bool <id>	BOOLEAN DECLARATION ENCOUNTERED. ID=<id>

bool <id> [<size>]	BOOLEAN DECLARATION ENCOUNTERED.
	ID=<id>
	SIZE=<size>

⟨location⟩ ⟨assign_op⟩ ⟨expr⟩	ASSIGNMENT OPERATION ENCOUNTERED
callout (⟨string_literal⟩ [, ⟨callout_arg⟩+,])	CALLOUT TO <string_literal> ENCOUNTERED
<location>	LOCATION ENCOUNTERED=<id>
<int_literal>	INT ENCOUNTERED=<int_literal>
<char_literal>	CHAR ENCOUNTERED=<char_literal>
<bool_literal>	BOOLEAN ENCOUNTERED=<bool_literal>
expr ‘+’ expr	ADDITION ENCOUNTERED
expr ‘-’ expr	SUBTRACTION ENCOUNTERED
expr ‘*’ expr	MULTIPLICATION ENCOUNTERED
expr ‘/’ expr	DIVISION ENCOUNTERED

expr ‘%’ expr	MOD ENCOUNTERED
expr ‘<’ expr	LESS THAN ENCOUNTERED
expr ‘>’ expr	GREATER THAN ENCOUNTERED

Evaluation details:
During manual evaluation, you will be required to show the codes (xyz.l and xyz.y) and output files (flex_output.txt and bison_output.txt). Be ready with sample test cases of your own.

Submission format:
Upload a zip file on moodle (before the evaluation date) containing the following files: 1) RollNo_Module1.l 2) RollNo_Module1.y 3) sample_test_pass.txt 4) sample_test_fail.txt 5) Readme 6) Makefile