

Department of Computer Engineering
University of Peradeniya
CO523 - Programming Languages

Lab 02: Syntax, Semantics, and Parsing

Lab Tasks

Task 1

Stratified Grammar:

```
<assignment-stmt> ::= <identifier> = <expr>;  
<expr>      ::= <expr> + <term> | <expr> - <term> | <term>  
<term>      ::= <term> * <factor> | <term> / <factor> | <factor>  
<factor>    ::= (<expr>) | <identifier> | <integer>  
<identifier> ::= a | b | c
```

Draw the parse tree for the following sentence using the **Stratified Grammar** above.

- I. $a = b + c * 10;$
- II. $c = 12 - (a * 5);$
- III. $b = (a / 3) + (2 * 5)$

Task 2

A recursive descent parser is a type of top-down parser that processes input using a set of mutually recursive functions, where each function corresponds to a grammar rule. The parser reads the input from left to right and constructs a parse tree by matching the grammar's production rules.

In this task, you are required to implement a recursive descent parser for simple arithmetic expressions using the C programming language.

Your implementation should correctly handle BODMAS operator precedence (Brackets, Orders, Division and Multiplication, Addition and Subtraction).

Download the provided skeleton code **Lab02_Skeleton.c** and use it as the starting point for your implementation.

Expected Output

After successful implementation, your program should behave as follows:

```
Enter a math expression (e.g., 2+3*4): 2+5*10
The result is: 52
```

```
Enter a math expression (e.g., 2+3*4): 3+(4*3)/2
The result is: 9
```

Submission

Create a ZIP file named CO523_Lab02_EXXYYYY.zip, where EXXYYYY represents your E-number. The ZIP file must contain the following items:

1. A PDF file containing the parse trees, which may be hand-drawn or digitally drawn, and are clearly readable.
2. The C source code file containing your implementation for Task 2.

Upload the Zip file to the FEeLS by the given deadline.