Syntax Analyzer

In this project, you will build a small compiler for MiniC language. In the second phase of this project, you are required to build a program that parse a stream of tokens and output a parse tree.

You should follow these steps to finish this phase:

1- Read tokens from the file.

2- Remove ambiguity in MiniJava grammar(if exist).

3- Prepare the types of each node that can be in the tree.

each node should include:

- Its own suitable member variables.
- Constructors to initialize these member variables.
- A function that pretty prints the class and its content.

4- Create a Parser class that implements MiniC grammar using Recursive descent parser.

This class should has:

A stack/Queue -or any suitable containers as you see fits- that holds tokens constructed by Lexical analyzer.

- A node root, which is the root of the constructed parse tree.
- A constructor.
- A parse function that calls the start rule in the grammar.
- Each grammar rule almost has a function.
- This function should return its suitable node data type from parse tree.

5- Test your parsing functions by printing the content of each node in the tree.

Notes:

• **The input** of this program is a stream of tokens read from a file.

• **The output** is a parse tree.

• **Grammar** is in a separate document named **MiniC**.

• **Submission should include:**

- Edited Token class(if any).

- Parsing code.

- Grammar after removing its ambiguity(if exist).

• Your submission will be on **Acadox** as zip or rar file.

• **Deadline:**

- First milestone for P2 part 1 16/4/2017
- First milestone for P2 part 2 23/4/2017
- First milestone for P2 part 3 30/4/2017
- Final submission 30/4/2017 at 11:55 PM

• **Name of the submitted file should follow** P2_Lab#_ID1_ID2_ID3_ID4.rar
Lab1 if you are from CS_IS_2 or CS_IS_2
Lab2 => CS_IS_3 or CS_IS_4 Lab3 =>
CS_IT or CS_DS