CSM2010 - Compiling Techniques
Course Assignment 2007-2008

Department of Computer Science. University of MALTA
Lecturer : Sandro Spina

The assignment is split in two parts. Part 1 (HTML Tags in Balance) carries 30% of the overall mark of the assignment while Part 2 (An implementation of a Shift Reduce Parser) carries 70% of the overall marks awarded to the assignment.

Part 1 – HTML Tags in Balance (30% of overall mark)

For the first part of the assignment you need to use the JFlex (http://jflex.de/) lexical analyser in order to implement a lexer for HTML pages. You will need to develop the specification file for JFlex and also possibly integrate Java code into it. Your program needs to check that for every opening tag there is a closing tag. In the case there is an imbalance in the tags your program should report which tag is not closed.

The following set of tags need to be returned as tokens by JFlex:

{HTML, HEAD, TITLE, BODY, H1, H2, P, BR, A, UL, LI, TABLE, TR, TD, TH}

Part 2 – A Shift Reduce Parser (70% of the overall mark)

For the second part of your assignment you need to design and implement a basic shift-reduce parser. You also need to write a grammar (which will then be parsed by your parser) which can unambiguously accept arithmetic (expr) and boolean (bexpr) expressions together with the basic statements :

if bexpr **then {** stmt **}**
id := expr
while (bexpr **) do { stmt }**
[stmt **;** stmt **]**

Programs in your language should have the form [stmt ; stmt ; ... ; stmt] which essentially would constitute a list (sequence) of statements.

Do not implement any back-tracking mechanisms in your parser. Also, you may safely assume that the grammars which will be fed into the parser are non-ambiguous. You will have to use JFlex classes in your program so that your parser can query for the next available token.

Note that in your grammar, multiplication and division should have higher operator precedence than addition and subtraction. Also brackets may be used to emphasize precedence. All boolean and arithmetic operators are left-associative.

Your parser should output a trace file indicating how it's parsing the programs in the grammar which you specify. For example it should specify which rule it's currently using and when the SHIFT and REDUCE operations are carried out. It should also notify the user whether the program syntax is valid or not. In the case it's not valid the parser should give an indication of the problem in the program to the programmer.