CS F214
Logic in CS
BITS Pilani, Hyderabad Campus
Assignment -2
Due Date : 17th October 2018 (by Midnight)
Total Marks: 30 (weightage : 10%)

Objective: This assignment is to help you understand that you can design a proof checking engine for rules of natural deduction. There are two parts of the assignment.

Part-I:

You have to implement the construction of a parse tree for a propositional logic formula. You can assume that the formula is well formed and fully parenthesized. The input to your program should be a propositional logical formula in infix notation. The connective symbols you will use for this assignment is as follows.

1. ~ for negation
2. V for OR
3. ^ for AND
4. > for implication.

Your code should have **separate functions** to perform the given tasks.

Task 1:

Write a function to **convert the infix** propositional logic expression **into a postfix** propositional logic expression. [5]

Task 2:

Write a function to **convert the postfix expression into a rooted binary parse tree**. [5]

Task 3:

Write a function to traverse the parse tree to **output the infix expression** back by in-order traversal of the parse tree. [5]

Once you have made a parse tree, you should be able to verify that a given formula is well formed.

Part II:

A tool for verifying whether a certain proof of a given sequent is valid or not. Allowed proof rules are :

1. Premise [1]
2. AND introduction/elimination [4]
3. OR introduction [5]
4. IMPLIES elimination [5]

Extra credit for : MT [3]

Definitions:

<statement> ::= p | ¬p | ¬(<statement>) | (<statement> ∧ <statement>) | (<statement> ∨ <statement>) | (<statement> → <statement>)

<rule> ::= ∧i | ∧e1 | ∧e2 | ∨i1 | ∨i2 | →e |P

Input:

First line:

n (number of statements)

Next n lines:

<statement>/<rule>[/line1[/line2]] (parameter in [] is optional whose existence will be determined by <rule>)

Output:

Valid Proof (or) Invalid Proof

Assumptions:

1. Line number starts from 1.
2. <statement> should be perfectly parenthesized, e.g. ((a ∧ b) ∧ c) is valid, (a ∧ b) ∧ c is invalid, ((a ∧ b)) is invalid, (a ∧ b) is valid, (a ∧ b ∧ c) is invalid, p is valid, (p) is invalid.
3. ¬ can be succeeded by a literal or '(' only.

Sample test case:

Input:

3

a/P

b/P

(a ∧ b)/∧i/1/2

Output: Valid Proof

General Instructions:

1. This assignment will be done in groups of max three students.
2. **Enter your team details in Google spreadsheet shared.**
3. **Code must be written in C/C++ language and gcc compiler only.**
4. The details of how to submit the code will be given later.
5. **You can discuss with your friends but refrain from copying the code and submitting. Also please do not use code downloaded from internet. Such codes will receive 0 credits.**
6. You have to demo the code to the instructor on a scheduled date and timing after submission. **It is important to attend the demo, as absence from demo will amount to no credit for the assignment.**

7. **Copied codes will receive no credits. No arguments will be entertained such as who copied fro whom. All involved parties will get 0 credit.**

References to look into:
http://scanftree.com/Data_Structure/infix-to-prefix

Algorithms in C++ by Robert Sedgewick