Assignment-2
CS F214
Logic in Computer Science

Total Weightage = 6% **Marks = 18**

General Instructions:

- The assignment is divided into three days. You need to create separate functions for each day.
- Function signatures and the main function will be given to you. You just need to write the function definitions and upload it.
- Your code will be tested with the given main function for each day.
- **You should strictly follow the filename nomenclature for each day.**
- For each day, the main file is named as “main_No.c” and the header file is named as “dayNo.h”. (eg: “main_1.c” and “day1.h”). You will create a new program file for each day and save it as “dayNo.c” (e.g. day1.c).
- If you fail to submit any task before the deadline, you can continue with the next task but you won’t be given marks for the task you didn’t complete.
- Include appropriate header files in your code whenever necessary.
- You should comment your code properly.
- **Assignment should be done sitting in Systems Lab or Data Science (I014 and I015) labs only.**
- 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.

Other Important Instructions:

- Please work as a team. There should be **only one submission per team**.
- **Do not share your code with other team members. Copied codes will be awarded zero marks for the entire assignment. Expecting all of you to be honest.**

Definition of Propositional Logic Formula-

<statement> ::= p | (¬p) |(~(<statement>)) | (<statement> Λ <statement>) | (<statement> V <statement>) |
(<statement> > <statement>)

Objective of the Assignment 2: Objective of this assignment is to implement the algorithm to convert any given propositional logic formula (in post-fix notation) into CNF form (in in-fix notation)

Day 1 (6th Nov 2019)

Marks = 5

Write a function ‘**impl_free**’ to remove implication from the given propositional logic formula.

Step 1: Make a parse tree from the given propositional formula by completing functions in ‘parse_tree.h’ and ‘stack.h’. (Refer to Assignment 1)
Step 2: Complete the functions given in ‘day1.h’ to remove implication from the parse tree.
Step 3: Complete the function for inorder traversal in ‘parse_tree.h’. (Refer to Assignment 1)
Note: Refer to lecture sessions for the algorithm to implement the function ‘impl_free’.

You will be given ‘main_1.c’, ‘day1.h’, ‘parse_tree.h’ and ‘stack.h’.

Create separate files - “day1.c”, “parse_tree.c” and “stack.c” where you will write all the function definitions.

During submission, you should include all the required header files, their respective definition files and one single driver code “main_1.c”.

Make sure in all files you have put all the group members’ name along with group ID.

Input -
A string containing postfix expression.

Output -
Implication free propositional logic formula.

Sample Test Case -

Input -
pq>

Output -
((~p)Vq)

Day 2 (7th Nov 2019)

(Marks = 6)

Complete the function in “day2.h” to get equivalent formula of the expression in NNF.

Complete the function signature given in “day2.h”.

The files given to you are “main_2.c”, “day2.h”, “stack.h”, “parse_tree.h” and “day1.h”.

Create a new file ‘day2.c‘ where you complete the functions given in ‘day2.h’.

During submission, you should include all the required header files, their respective definition files and one single driver code “main_2.c”.

Input -
A string containing postfix expression.
Output -
Propositional logic formula’s equivalent form in NNF.

Input -
p~q>
Output -
(pVq)

Day 3 (8th Nov 2019)
(Marks = 7)

Today's objective is to convert a given postfix expression into its equivalent formula in CNF .

Complete the function signature given in **'day3.h'**.

The files given to you are **'main_3.c', day3.h', 'stack.h', 'parse_tree.h', 'day1.h' and 'day2.h'**.

Create a new file **'day3.c'** where you complete the functions given in 'day3.h'.

During submission, you should include **<u>all the required header files, their respective definition files</u>** and one single driver code "main_3.c".

Input -
String containing a postfix expression.

Output-
Equivalent formula in CNF.

Sample Test Case -
Input -
p~q>r^
Output -
((pVq)^r)