Assignment-1
CS F214
Logic in Computer Science

Total Weightage = 8% Total Marks = 24

General Instructions:

- The assignment is divided into four 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> ∧ <statement>) | (<statement> ∨ <statement>) |
(<statement> > <statement>)

Day 1 (14th Oct 2019)

Marks = 4

Learn how to initialize a stack to store characters. Implement the follow operations in stack through separate functions:

1. push(x)
2. pop()
3. isEmpty()
4. isFull()
5. top()

Your code should comply with the main function.

The filename of the header file will be "day1.h". The "day1.h" code is given to you. Please use these function signatures only.

Create a separate file - "day1.c" where you will write all the function definitions.

The "main_1.c" is the main file to test your code. Use it to test your code.
Make sure in all files you have put all the group members' name along with group ID.

Input -

MAX, the first line contains the maximum number of elements in the stack.
N, the second line contains the number of operations user wants to execute.
The next N lines contain various choice of operations ranging from 1 to 5 as mentioned above.
For push operation, your code should take a character as input.

Output -

Your function should generate appropriate outputs for isEmpty(), isFull() and top() functions.

Sample Test Case -

Input -

```
10
8
1 a
1 b
1 c
2
3
4
5
1 d
```

Output -

```
false
false
b
```

Day 2 (15th Oct 2019)

Marks = 6

The task of Day 2 is to be able to convert a well-defined fully parenthesized propositional logic formula given in infix notation to convert it into post-fix notation.

Infix notation means that the operator is written between two operand for example p ^ q.
Post fix notation means the operator is written after the two operands for example above formula can be written as pq^.

Now that you have learnt how to use a stack, implement the following functionality using stack.

Write a function to convert an infix expression to a postfix expression. The function signature is given in the header file "day2.h" :

You should name your program code as "day2.c" where you will complete the function definition.

The main_2.c is the file to test your code.

Input -
A string containing a well formed infix expression.
Output -
Postfix expression.

Sample Test Case -
Input -
((pVq)^r)
Output -
pqVr^

Algorithm for converting from infix to postfix notation.
1. Scan the infix expression from left to right.
2. If the scanned character is an operand, output it.
3. If the scanned character is an operator, push it into the stack.
4. If the scanned character is '(', push it into the stack.
5. If the scanned character is an ')' , output the stack and pop until a '(' is found.
6. Repeat the steps until scanning is finished.
7. Pop out all the leftover elements in the stack.
8. You will have to include "day1.h" to use stack operations.