Aniruddh N.S
1BM18CS015
29/12/2020

AI Lab Test - 2

Writeup :-

(2) Consider P, Q & R as variables & the knowledge base contains following sentences:
$(P \wedge Q) \Rightarrow R$; $(Q \Rightarrow P)$; Q
Design code for TT entailment & show whether KB entails R.

```
import numpy as np
import os
import pandas as pd

Combinations = [(True, True, True), (True, True, False),
(True, False, True), (True, False, False),
 (False, True, True), (False, True, False),
 (False, False, True), (False, False, False)]
variables = {'P': 0, 'Q': 1, 'R': 2}

 kb = ''
 q = ''
 priority = {'~': 3, 'v': 1, '^': 2}

 def input_rules():
     global kb, q
     kb = input("Enter the rule: ")
     q = input("Enter query: ")
```

```
def entailment():
    global kb, q
    print ("*"*10 + "Truth Table Reference"
            + "*"*10)
    print ("P  ", "  Q  ", "  R  ", "  KB", "Alpha")
    print ("*"*20)
    for comb in combinations:
        s = evaluatePostfix (toPostfix (kb) comb)
        f = evaluatePostfix (toPostfix(q), comb)
        a,b, c = comb
        print (a,b, c,s,f)
        print ("-"*10)
        if s and not f:
            return False
    return True
def isOperand (c):
    return c.isalpha () and c!='v'
def isLeftParenthesis (c):
    return c == '('
def isRightParenthesis (c):
    return c == ')'
def isEmpty (stack):
    return len(stack)==0
def peek (stack):
    return stack[-1]
def hasLessOrEqualPriority (c1, c2):
    try: return priority [c1] <= priority [c2]
    except KeyError: return False
def toPostfix (infix):
    stack=[]
    postfix=''
    for c in infix:
        if isOperand(c):
            postfix+= c
        else:
            if isLeftParenthesis(c):
                stack.append(c)
            elif isRightParenthesis(c):
                operator = stack.pop()
                while (not isLeftParenthesis(
                                        operator)):
                    postfix += operator
                    operator = stack.pop()
            else:
                while (not isEmpty(stack))
                  and hasLessOrEqualPriority
                        (c,peek(stack)):
                    postfix += stack.pop()
        return postfix
def evaluatePostfix(exp, comb):
    stack=[]
    for i in exp:
        if isOperand(i):
            stack.append(comb[variable[i]])
        elif i == "~":
            val1 = stack.pop()
            Stack.append(not val1)
        else:
            val1 = stack.pop()
            val2 = stack.pop()
    return stack.pop()
input_rules()
ans = entailment()
if ans: print("Entails")
else: print("Doesnot entail").
```