Pagaar Mittal

1BM18CS075

B2

AI-LAB TEST - 2

Question - 3

Consider P, Q and R as variables and the knowledge base contains following sentence.

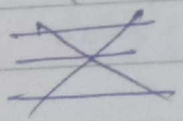
$Q \Rightarrow R$, $P \Rightarrow \neg Q$; $R \vee Q$

Design the code for TT entailment and show whether knowledge base entails R.

Code :

```
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)]

variable = {'P': 0, 'Q': 1, 'R': 2}
    kb = ''
    q = ''
priority = ('~': 3, 'v': 1, '^': 2)
```



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

def entailment ();
```

```
global Kb, q
print ('' * 10 + "Truth Table Reference" + '' * 10)
print ('Kb', 'alpha')
print ('*' * 10)
for comb in combinations:
    s = evaluatePostfix (toPostfix (Kb), comb)
    f = evaluatePostfix (toPostfix (q), comb)

    print (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 hasLessOrEqual
                        Priority (c, peek (stack)):
                    postfix += stack.pop ()
                stack.append (c)
    while (not isEmpty (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()
        val1 = stack.pop()
        stack.append(_eval(i, val2, val1))
    return stack.pop()
def _eval(i, val1, val2):
    if i == '^':
        return val2 and val1
    return val2 or val1

input_rules()
ans = entailment()
if ans:
    print("The knowledge Base entails query")
else
    print("The knowledge Base does not
                entail query")
```

Roshan