Ai-Lab-2

Rajath. M.K
1BM18CS079
5-B sec.

Q) Consider P, Q & R as variables & knowledge base contains following:

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

Entailment code:

```
combi = [(True, True, True), (False, True, True)
         (True, True, False), (False, True, False
         (True, False, True), (False, False, True)
         (True, False, False), (False, False, False)]

variable = {'p': 0, 'q': 1, 'r': 2}
priority = {'~': 3, 'v': 1, '^': 2}

def input_rules():
    global kb, q
    kb = (input("Enter Rule: "))
    q = (input("Enter Query: "));
```


Byt..

```
def toPostfix(infix):
    stack = []
    postfix = ''
    for c in infix:
        if isOperand(c):
            postfix += c
        else
            if isLeftParanthesis(c):
                stack.append(c)
            elif isRightPar(c)
                operator = stack.top()
                while not isLeftParanthesis(operator):
                    postfix += operator
                    operator = stack.pop()
                stack.append(c)
            while (not isEmpty(stack)):
                postfix += stack.pop()
            return postfix
```

```
def eval(i, val1, val2):
    if i == '^': return val2 and val1
    return val2 or val1

def evalPostfix(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()
            stack.append(eval(i, val2, val1))
    return stack.pop()

def isOperand(c):
    return c.isalpha() and c != 'v'

def isleftPeran(c):
    return c == '('

def isRightPar(c)
    return c == ')'
```


Repo.

```
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 entailment():
    global kb, q
    print("kb", "alpha")
    print("-" * 7)
    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
```

```
input_rules()
ans = entailment()
# (~q v r) ~ (~p v ~q) ^ (r v q) #Rule

if ans : print("Knowledge base entails query")
else : print("knowledge base doesnt entail")
```

5

Output..