17/1/2022.

AI LAB TEST-2.

Disha B.
1BM19CS050
5'A'

1). Write a program to create a knowledge base using propositional logic and show that the given query entails the knowledge base or not.

```
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
```

Disha B.
1BM19CS050

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

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

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

def isEmpty(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 isLeftParanthesis(c):
                stack.append(c)
            elif isRightParanthesis(c):
                operator=stack.pop()
                while not isLeftParanthesis(operator):
                    postfix+=operator
                    operator=stack.pop()
            else:
                while(not isEmpty(stack)) and
                    hasLessOrEqualPriority(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()
            val2 = 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 doesn't entail
                                    query")
```

expected output: there when KB is true, if x is also true only then KB⊨x.