Date: 28/12/2020 AI-Lab-2 Name: Shreya Laddha
USN: 1BM18CS103

(2)

Question: Consider S and T as variable and the following represents the relationships:

(a) $\neg(S \vee T)$

(b) $T \vee \neg T$

Code:

```
Variable = {'S': 0, 'T': 1}

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

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

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

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

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

def isEmpty(stack):
    return len(stack) == 0
```

Name: Shreya Laddha
USN: 1BM18CS103

```
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 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()
```

Name: Shreya Laddha
USN: 1BM18CS103

```
def checkEntailment():
    kb = [input("Enter KB : ")]
    query = [input("Enter query:")]
    combinations = [ [True, True], [True, false ], [false, True]
                     [false, false]]

    postfix_kb = toPostfix(kb)
    postfix_q = toPostfix(query)
    for combination in combinations:
        eval_kb = evaluatePostfix( postfix_kb, combination)
        eval_q = evaluatePostfix( postfix_q, combination)
        print(combination, ": kb = ", eval_kb, ": q = ", eval_q)
        if(eval_kb == True):
            if(eval_q == false):
                print("does not entail !! ")
            print ("Entails")

checkEntailment()
```