Samudhi.M
IBM18CS092
B-3

AI Lab Test-1

```
variable = {'p': 0, 'q': 1, 'r': 2}.
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 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
```


Samudhi.M

Samudhi. M
1BM18CS092

```
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
```

Samudhi. M
1BM18CS092

```
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 checkEntailment():
    kb = (input("Enter the knowledge base: "))
    query = (input("Enter the query: "))
    combinations = [[True, True, True],
                    [True, True, False],
                    [True, False, False True],
                    [True, False, False],
                    [False, True, True]
                    [False, True, False],
                    [False, False, True],
                    [False, False, False]]
    postfix_kb = toPostfix(kb)
    postfix_q = toPostfix(query)
```


Samudhi. M

```
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("doesn't entail!!")
            return False

print("Entails")

if __name__ == "__main__":
    CheckEntailment().
```