Harshit Hiremath
IBM18CS036
29/12/2000
5A2

AI Lab Test 2

Q2.:- Given rules

$P \wedge Q \Rightarrow R$, $Q \Rightarrow P$, Q.

=> 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)]

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

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 hasLessEqualPriority(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 input_rules():
    global kb, q
    kb = input("Enter rule: ")
    q = input("Enter query: ")
```

Harshit Hiremath
1BM18CS036

```
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
```