Q. 2

Saman Kumar Pal
IBM CSE III
Date: / /

AI lab Git 2

Combination :

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 query"))
def eval(i, val1, val2):
    if i == '^':
        return val2 & val1
    return val2 or val1
def evaluatePostfix(exp, Count)
    stack = []
    for i in exp:
        if isOperand(i)
            stack.append(Count[i])
        elif i == '~':
            val1 = stack.pop()
            stack.append(not val1)
    else
        val1 = stack.pop()
        val2 = stack.pop()
        stack.append(eval(i, val1, val2))
```

Date: / /

```
return Stack.pop()
def toPostfix(infix):
    stack = []
    postfix = ''
    for c in infix:
        if isOperand(c):
            postfix += c
        else:
            if isLeftParenthesis(c):
                stack.append
            elif isRightParenthesis(c)
                operator = stack.pop()
        while not isLeftParenthesis()
        postfix += operator
        operator = stack.pop()
    else
        while (not isEmpty(stack)
            & hasLessOrEqualPriority
    postfix += stack.pop()
        stack.append(c)
    while (not isEmpty(stack)):
        postfix += stack.pop()
    return postfix.
```

```
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

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

def peek(stack):
    return stack[-1]

def hasLessOrEqualPriority(c1, c2):
    try: return priority[c1] <= priority[c2]
    except KeyError: return False
```

Date: /

```
input_[illegible],
    ans = entailment()
        if ans:
            print('Knowledge base entails
                          query')
    else
        print('Knowledge base does
                   not entail query)
```