Artificial Intelligence
lab test-2.

Pramod D.Y
1BM19CS405
01/01/2021
Pramod D.Y

Consider P, Q1 and R as variable and the knowledge base contains following sentences

$Q1 \Rightarrow R$; $P \Rightarrow \neg Q$; $R \vee Q$.

Design the code for TT entailment and show whether knowledge base entails R.

Code :-

```
In [1]:
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}

In [2]:
def input_rules():
    global kb, q
    kb = input("Enter Rule")
    q = input("Enter Query")
```

q = (input (" Enter Query:"))

In [3]:

```
global kb, q
print ("*" * 10 + " Truth table Reference " +
                "*" * 10)
print ("kb", "alpha")
print ("-" * 10)
for comb in combinations:
    s = evaluate Postfix (to Postfix (kb), comb)
    q = Evaluate Postfix (to Postfix (q), comb)
    print (s, q)
    print ("-" * 10)
    if s and not q:
        return False
return True
```

In [4]

```
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 prec[c1] <= prec[c2]
    except KeyError: return False

in[5]
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()
            while (not isEmpty(stack)) and
                hasLessOrEqualPriority(c, peek(stack)):
                    postfix += stack.pop()
```

Pramod Dey

Pramod Dey

```
etre etack.append(c)
while (not peEmpty(stack)):
    postfix += stack.pop()
    return postfix
```

In [6]:

```
def _eval (i val1, val2):
    if i == '^' return val2 and val1
    return val2 or val1
def evaluatePostfix (exp, comb):
    Stack = []
    for i in exp:
        stack.append (comb[variable[i]])
        elif i == '-':
            stack.append (not val1)
        else:
            val1 = stack.pop()
            val2 = stack.pop()
            stack.append (_eval (i, val2, val1))
    return stack.pop()
```

In [11]:

```
Input_values()
ans = entailment()
```


Pramod D Y

IBM19CS405
Pranod

of ans:

print ("the knowledge base Entails Query")

else:

print ("the knowledge base Does not Entail Query")

Pranodh