Artificial Intelligence
lab test-2

Pramod D.Y
1BM19CS405
01/01/2021
Program 9

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

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

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

Code :-

```
combinations = [(True, True, True), (True, True,
False), (True, False, True), (True, False, False),
(False, True, True), (False, True, False), (False,
True), (False, False, False)]
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")
```

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

[redacted]

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

[redacted]

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


Pramod D.Y.

else stack.append(c)

while (not isEmpty(stack)):

postfix += stack.pop()

return postfix

[redacted]

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 == '-1':

stack.append (not val1)

else:

val1 = stack.pop()

val2 = stack.pop()

stack.append (_eval (i, val2, val1))

return stack.pop()

[redacted]

Input_values()

ans = entailment()

Bamod

IBM19CS405
Pranod

pf ans:

print ("the knowledge base Entail Query")

else:

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

Pranodf