AI Lab test 2.

Akanksha Reddy.

1BM18CS007.

Date:
P. No:

Batch B1.

Section 5A.

Date: December 29, 20.

Program :- To design code for TT entailment and show whether knowledge base entails query or not.

Code :-

```
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]
```

mean

Date:
P. No:

```
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 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 hasLessorEqualPriority(c,peek(stack)):
                    postfix += stack.pop()
                stack.append(c)
    Akanksha:
```

Date:
P. No:

```
while (not isEmpty(stack)):
        postfix += stack.pop()

    return postfix

def evaluatePostfix(exp, conv):
    Stack = []
    for i in exp:
        if isOperand(i):
            stack.append(conv[variable[i]])
        elif i == 'n':
            val1 = stack.pop()
            stack.append(not val1)
        else:
            val1 = stack.pop()
            val2 = stack.pop()
            Stack.append(_eval(i, val1, val2))

    return stack.pop()

def checkEntailment():
    kb = (input("enter knowledge base"))
    query = (input("enter query: "))
    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]]
```

Akanksha

Date:
P. No:

```
poltfix_kb = toPoltfix(KB)
poltfix_q = to Poltfix(query)
for combination in combinations:
        eval_kb = evaluate Poltfix(poltfix_KB,
                                   combination)
        eval_q = evaluate Poltfix(poltfix_q,
                                  combination)
print(combination, ": kb =", eval_kb,
  ": q =", eval_q)

if (eval_kb == True):
        if (equal_q == false):
                print("Doesn't entail")
                return False

        print("entails")

checkEntailment().
```