AI - Lab test 2

(2)

IB21CS002
Abhijnya K.G
V^th sem, CSE
29/12/20

Pseudocode:-

```
global Kb, q
def start():
    Kb = (input("Enter the Kb statement - "))   # Kb = R ∧ ~Q   (given)
    q = (input("Enter the query - "))           # q = R

def ~~entailment~~():
    ~~for c in~~
    # as there are 3 variables P, Q, R
    combinations = [(True, True, True),
                    (True, True, False),
                    (True, False, True),        # 2^3 = 8 combination
                    (True, False, False),
                    (False, True, True),
                    (False, True, False),
                    (False, False, True),
                    (False, False, False)]

def entailment():
    for c in combinations:
        s = postfixEvaluation(convert_to_postfix(Kb), c)
        f = postfixEvaluation(convert_to_postfix(Kb), c)
        print(s, f)
        if s and not f:
            return false

return True
```


Abhi

```
# convert to postfix to evaluate the expression
def convert_to_postfix(infix):
    stack = []
    for c in infix:
        if (isoperand(c)):
            postfix = postfix + c
        else:
            if (c = '('):
                stack.append(c)
            elif (c = ')'):
                operator = stack.pop()
                while not operator = '(';
                    postfix += operator
                    operator = stack.pop()
            else:
                while (not isEmpty(stack))
                    postfix += stack.pop()
                stack.append(c)
    while (not isEmpty(stack)):
        postfix += stack.pop()
    return postfix.

def postfixEvaluation(exp, comb):
    stack = []
    for i in exp:
        if isoperand(i):
            stack.append(comb[variable[i]])
        elif i == '~':
            val1 = stack.pop()
            stack.append(not val1)
```


Abi

```
    else:
        val1 = stack.pop()
        val2 = stack.pop()
        stack.append(eval(i, val1, val2))
return stack.pop()

def eval(i, val1, val2):
    if i == '^':
        return val2 and val1
    return val2 & val1

start()
result = entailment()
if result:
    print("KB entails");
else:
    print("KB does not entails");
```

Given:- $(P \wedge Q) \Rightarrow R$ $\quad Q \Rightarrow P$ $\quad Q$

Conversion to CNF:

$\neg(P \wedge Q) \vee R$ $\quad \neg Q \vee P$ $\quad Q$

$\neg P \vee \neg Q \vee R$ $\quad \neg Q \vee P$ $\quad Q$

$(\neg P \vee \neg Q \vee R) \wedge (\neg Q \vee P) \wedge Q$

$\neg P \vee \neg Q \vee R \wedge \neg Q \vee P \wedge Q$

$R \wedge \neg Q$

$\Rightarrow R \wedge \sim Q$