Arfath Kabir 1BM18CS130

2)

```
variable = {'s' : 0, 'T' : 1}
Priority - {'~' : 3 ,'^' : 1 , 'v' : 2}
def eval (i, val1, val2) :
if i == '^':
    return val2 and val1
    return val2 or val1
def is operand (c):
        return c. is alpha() and (c1 = 'v')
def is left Parantheris (c):
        return c == '('
def isRight Parantheris (c):
        return c == ')'
def is Empty (stack):
        return len (stack) == 0
def peek (stack):
        return stack[-1]
def has less or Equal Priority (c1, c2)
    try:
        return priority [c1] <= priority [c2]
    except KeyError:
        return False.
def to Postfix (infix):
    stack = []
    post fix = ''
    for c in infix:
        if isoperand (c):
            postfix += c
        else
            if is left Parantheris (c):
                stack. append (c)
```

```
elif isRightParanthesis(c):
    stack.append(c)
elif isRightParanthesis(c):
    operator = stack.pop()
    while not isLeftParanthesis(operator):
        postfix += operator
        operator = stack.pop()
    else
        while (not isEmpty(stack)) and
        has less or Equal Priority(c, peek(stack))
        stack.append(c)
while (not isEmpty(stack)):
    postfix += stack.pop()
    return postfix.

def evaluatePostfix(exp, comb):
    stack = []
    for i in exp:
    else
        val1 = stack.pop()
        stack.append(not val1)
    else.
        val1 = stack.pop()
        val2 = stack.pop()
        return stack.pop()

def CheckEntailment()
    kb = (input "Enter the knowledge base:")
    query = (input "Enter the query")
    combination = [(True, True),
                   (True, False),
                   False, True), (False, False)]
```

```
for combination in combinations
    eval_kb = evaluatePostfix(postfix_kb, comb)
    eval_q = evaluatePostfix(postfix_q, combination)
    print(combination, "kb=", eval_kb, "q=", eval_q)
    if (eval_kb == True):
        if eval_q == False):
            print("Doesnt entail")
            Return False
    print("Entails")

checkEntailment()
```