**NAME:** NILANSHU

**USN:** 1BM18CS062

**DATE:** 20-11-2020

**LAB 4**: Create a knowledgebase using prepositional logic and show that the given query entails the knowledge base or not.

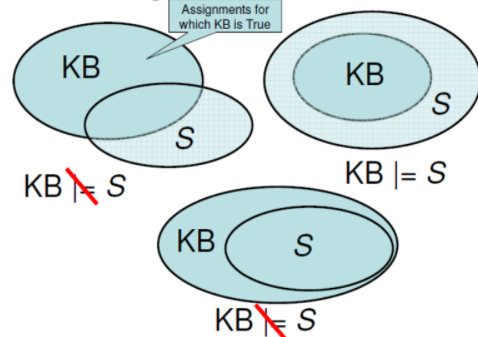
**Explanation:**

Knowledge Base (KB): a collection of sentences of a formal language.

* Model: an assignment of (True/False) values to each of the symbols. If the knowledge base is built from n symbols, there are 2n possible models.
* Evaluation: A sentence s is evaluated on a model m by setting each symbol to its corresponding value in m. The result of the evaluation is a value in {True,False}
* KB Evaluation: The result of the KB evaluation is the conjunction of the results of the evaluations of all the sentences in KB

**Rules for entailment:**

* “KB entails S” if all the models that evaluate KB to true also evaluate S to True.
* We do not care about those models that evaluate KB to False. The result of evaluating S for these models is irrelevant.

****

**Implementation of code:**

* Declaring variables, agents and all models of KB

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}

* Getting input: KB and S

def input\_rules():

* Checking for entailment

def entailment():

* Converting to postfix

def toPostfix(infix):

* Comparing all cases with the sentence

def evaluatePostfix(exp, comb):

* Returning and of two values

def \_eval(i, val1, val2):

* Checking for priority for converting into postfix and comparing with sentence models

def isOperand(c):

def isLeftParanthesis(c):

def isRightParanthesis(c):

def isEmpty(stack):

def peek(stack):

def hasLessOrEqualPriority(c1, c2):