Program 6: Create a knowledgebase using prepositional logic and show that the given query entails the knowledge base or not .

Write\_up:

_	DONAL TO OLIVE
	PRANAV JAGADEESH IBMIRCSO71 PROGRAM-6
	combinations = [(True, True, True), (True, True, False),
	( sull, False ) take, (False, True, True),
	(False, Tome, Trues), (False, Tome, False)
	(False, False, True), (False, False, False)
	Variable = { 'p':0, 'a':1, 'n':23
	Kb = ''
	$Q = \frac{1}{2}$
	phiority = { (~ ': 3) [v/: 1] 1/2 23
	def input_ruler():
	alolel Kb, a
	Kb = (inbut (" (-ntes rule: "))
	of = l'iprit ("Enter the Query:")
	def antailment():
	dulial Kh a
	point (' ' + 10 +" Fouth Table Beference " + " * 10)
	print ('Kb'; (alpha')
	print (1 x 1x10)
	for comb in combinations:
	s = evaluate Portfin (toPortfin (Kb), comb)
	f = evaluate Portfix (to Portfix (9), comb)
	point (s, f)
	Frunt ( - 1 + 10)
	if 5 and not f:
	return false
	return True
	def is Operand (c):
	neturn & realpho() and c = V
	def is left Parantheris (c):  return c == 1 (')
	30to 600 C == 1 C)
	dol in Right Paramble in (c)
	def ix Right Parantheris (c); neturn c = = ()
	Juka ove C

def ist mpty (stack):

return len (etack) == 0

def feek (stack):

neturn stack (-1)

def harlen Ortqual Priority (c1, (2): return priority [c1] <= priority [c2]
except Key trush; def to Portfix (infix): ile not is/eft Parantheris (Operator poetfeix += operator else : e (not is Empty (stack)) and has Les Os Equal Porcerity (c, peck [stack)): etack pop

while (not is fropty (stack)):

'pertfix t = stack bob() def enaluate Poetfix (exp, combe): vtack = [] for i in exp:

if is operand (i):

stack append (combe Evaviable [i]])

elef i == (~': vall = stack.fop() Mack.append(not vall) stack affend (enal (e, nal 2, val 4 ans = entilpment () else: print ("The Knowledge Base entaile query")
else: print ("The Knowledge Base does not entail query")

```
Program:
```

```
combinations=[(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}
def input_rules():
  global kb, q
  kb = (input("Enter rule: "))
  q = input("Enter the Query: ")
def entailment():
  global kb, q
  print("*10+"Truth Table Reference"+"*10)
  print('kb','alpha')
  print('*'*10)
  for comb in combinations:
     s = evaluatePostfix(toPostfix(kb), comb)
     f = evaluatePostfix(toPostfix(q), comb)
     print(s, f)
     print('-'*10)
     if s and not f:
       return False
  return True
def isOperand(c):
  return c.isalpha() and c!='v'
def isLeftParanthesis(c):
  return c == '('
def isRightParanthesis(c):
```

```
return c == ')'
def isEmpty(stack):
  return len(stack) == 0
def peek(stack):
  return stack[-1]
def hasLessOrEqualPriority(c1, c2):
  try:
     return priority[c1]<=priority[c2]</pre>
  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()
       else:
          while (not isEmpty(stack)) and hasLessOrEqualPriority(c, peek(stack)):
            postfix += stack.pop()
```

```
stack.append(c)
  while (not isEmpty(stack)):
     postfix += stack.pop()
  return postfix
def evaluatePostfix(exp, comb):
  stack = []
  for i in exp:
     if isOperand(i):
       stack.append(comb[variable[i]])
     elif i == '~':
       val1 = stack.pop()
       stack.append(not val1)
     else:
       val1 = stack.pop()
       val2 = stack.pop()
       stack.append(_eval(i,val2,val1))
  return stack.pop()
def _eval(i, val1, val2):
  if i == '^':
     return val2 and val1
  return val2 or val1
input_rules()
ans = entailment()
if ans:
  print("The Knowledge Base entails query")
else:
  print("The Knowledge Base does not entail query")
input_rules()
ans = entailment()
```

```
if ans:
 print("The Knowledge Base entails query")
else:
 print("The Knowledge Base does not entail query")
Output:
                Enter rule: (~qv~pvr)^(~q^p)^q
                Enter the Query: r
                Truth Table Reference
                kb alpha
                *****
                False True
                -----
                False False
                -----
                The Knowledge Base entails query
                Enter rule: (pvq)^(~rvp)
                Enter the Query: r
                Truth Table Reference
                kb alpha
                *****
                True True
                -----
                True False
                -----
                The Knowledge Base does not entail query
                Process finished with exit code 0
```