Program 7 : Knowledge Base Resolution

Code:
```
def tell(kb, rule):
    kb.append(rule)


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)]



def ask(kb, q):
    for c in combinations:
        s = all(rule(c) for rule in kb)
        f = q(c)
        print(s, f)
        if s != f and s != False:
            return 'Does not entail'
    return 'Entails'



kb = []

# Get user input for Rule 1
rule_str = input("Enter Rule 1 as a lambda function (e.g., lambda x: x[0] or x[1] and (x[0] and
x[1]): ")
r1 = eval(rule_str)
tell(kb, r1)

# Get user input for Query
query_str = input("Enter Query as a lambda function (e.g., lambda x: x[0] and x[1] and (x[0] or
x[1]): ")
q = eval(query_str)

# Ask KB Query
result = ask(kb, q)
print(result)
```
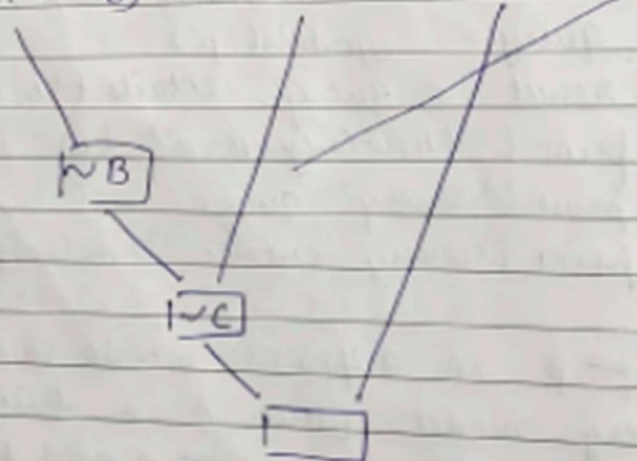
Date : 29/12/23

Knowledge Based Resolution

```
def negate_literal(literal)
    if literal[0] == 'N'
        return literal[1:]
    else
        return 'N' + literal

def resolve(c1, c2)
    resolved_clause = set(c1) | set(c2)
    for literal in c1
        if negate_literal(literal) in c2:
            resolved_clause.remove(literal)
                                    (negate_literal
    return tuple(resolved_clause)  (literal))
```

$LB : (A \vee \sim B) \wedge (B \vee \sim C) \wedge C \wedge \sim A$

Output:

```
Kanjika Singh-1BM21CS086

Step      |Clause |Derivation
-----------------------------
 1.       | Rv~P  | Given.
 2.       | Rv~Q  | Given.
 3.       | ~RvP  | Given.
 4.       | ~RvQ  | Given.
 5.       | ~R    | Negated conclusion.
 6.       |       | Resolved Rv~P and ~RvP to Rv~R, which is in turn null.
A contradiction is found when ~R is assumed as true. Hence, R is true.
```

```
Kanjika Singh-1BM21CS086

Step      |Clause |Derivation
-----------------------------
 1.       | PvQ   | Given.
 2.       | ~PvR  | Given.
 3.       | ~QvR  | Given.
 4.       | ~R    | Negated conclusion.
 5.       | QvR   | Resolved from PvQ and ~PvR.
 6.       | PvR   | Resolved from PvQ and ~QvR.
 7.       | ~P    | Resolved from ~PvR and ~R.
 8.       | ~Q    | Resolved from ~QvR and ~R.
 9.       | Q     | Resolved from ~R and QvR.
10.       | P     | Resolved from ~R and PvR.
11.       | R     | Resolved from QvR and ~Q.
12.       |       | Resolved R and ~R to Rv~R, which is in turn null.
A contradiction is found when ~R is assumed as true. Hence, R is true.
```

```
Kanjika Singh-1BM21CS086

Step      |Clause |Derivation
--------------------------------
 1.       | PvQ   | Given.
 2.       | PvR   | Given.
 3.       | ~PvR  | Given.
 4.       | RvS   | Given.
 5.       | Rv~Q  | Given.
 6.       | ~Sv~Q | Given.
 7.       | ~R    | Negated conclusion.
 8.       | QvR   | Resolved from PvQ and ~PvR.
 9.       | Pv~S  | Resolved from PvQ and ~Sv~Q.
10.       | P     | Resolved from PvR and ~R.
11.       | ~P    | Resolved from ~PvR and ~R.
12.       | Rv~S  | Resolved from ~PvR and Pv~S.
13.       | R     | Resolved from ~PvR and P.
14.       | S     | Resolved from RvS and ~R.
15.       | ~Q    | Resolved from Rv~Q and ~R.
16.       | Q     | Resolved from ~R and QvR.
17.       | ~S    | Resolved from ~R and Rv~S.
18.       |       | Resolved ~R and R to ~RvR, which is in turn null.
A contradiction is found when ~R is assumed as true. Hence, R is true.
```