KB - resolution

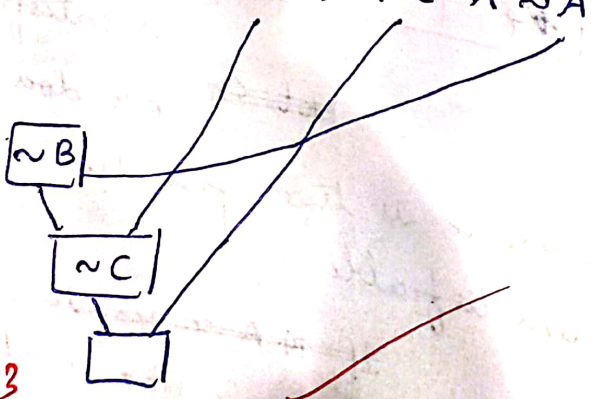
```
def negate_literal(literal):
    if literal[0] == '~':
        return literal[1:]
    else
        return '~' + literal

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(literal))
    return tuple(resolved_clause)

def resolution(kB)
    new_clauses = set()
    for i, c1 in enumerate(kB)
        for j, c2 ——— (kB)
            if i != j:
                new_clause = resolve(c1, c2)
                if len(new_clause) > 0 & new_clause not in
                    new_clauses.add(new_clause)
```

Output:

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



29/12/23

```
79    rules = 'Rv~P Rv~Q ~RvP ~RvQ' #(P^Q)<=>R : (Rv~P)v(Rv~Q)^(~RvP)^(~RvQ)
80    goal = 'R'
81    main(rules, goal)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\neha2\OneDrive\Documents\NehaKamath_1BM21CS113_AILab> python -u "c:\Us

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.
PS C:\Users\neha2\OneDrive\Documents\NehaKamath 1BM21CS113 AILab>
```