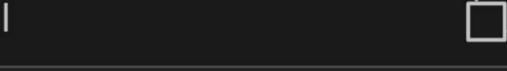
```
KB_Resolution.py △
2 def negate(term):
             return f' \sim \{term\}' if term[0] != '\sim' else term[1]
5 def reverse(clause):
             if len(clause) > 2:
                      t = split_terms(clause)
10 def split_terms(rule):
             exp =
             terms = re.findall(exp, rule)
               return terms
14 def contradiction(query, clause):
15    contradictions = [ f'{query}v
                                                                               [negate(query)}', f'{negate(query)}v{query}']
             return clause in contradictions or reverse(clause) in contradictions
            resolve(kb, query):
             temp = kb.copy()
             temp += [negate(query)]
20
             steps = dict()
             for rule in temp:
                      steps[rule] =
             steps[negate(query)] = 'Negated conclusion.
             while i < len(temp):
                      n = len(temp)
                      j = (i + 1) \% n
                      clauses = []
29
30
                               terms1 = split_terms(temp[i])
                               terms2 = split_terms(temp[j])
                               for c in terms1:
                                        if negate(c) in terms2:
                                                t1 = [t for t in terms1 if t != c]
t2 = [t for t in terms2 if t != negate(c)]
34
35
                                                gen = t1 + t2
                                                 if len(gen) == 2:
                                                         if gen[0] != negate(gen[1]):
38
                                                                 clauses += [f'{gen[0]}v{gen[1]}']
39
40
                                                                 if contradiction(query, f' {gen[0]}v{gen[1]}'):
42
                                                                           temp.append(f
                                                                           steps[ \ ] = f"Resolved {temp[i]} and {temp[j]} to {temp[-1]}, which is in turn number of temp[-1] and temp[-1] to {temp[-1]}, which is in turn number of temp[-1] and temp[
                                                                           \nA contradiction is found when {negate(query)} is assumed as true. Hence, {query
44
                                                                           return steps
46
                                                elif len(gen) == 1:
                                                         clauses += [f'{gen[0]}']
47
48
                                                         if contradiction(query, f'{terms1[0]}v{terms2[0]}'):
   temp.append(f'{terms1[0]}v{terms2[0]}')
   steps[''] = f"Resolved {temp[i]} and {temp[j]} to {temp[-1]}, which is in turn null.
49
50
                                                                   \nA contradiction is found when {negate(query)} is assumed as true. Hence, {query} is
                                                                  return steps
54
                               for clause in clauses:
                                        if clause not in temp and clause != reverse(clause) and reverse(clause) not in temp:
                                                 temp.append(clause)
                                                 steps[clause] = f'Resolved from {temp[i]} and {temp[j]}
                               j = (j + 1) \% n
                      i += 1
60
              return steps
61 def
             resolution(kb, query):
             kb = kb.split(' '
63
             steps = resolve(kb, query)
             64
65
             i = 1
67
              for step in steps:
68
                      print(f' {i}.\t| {step}\t| {steps[step]}\t')
70 def main():
             kb = input()
             print("Enter the
query = input()
             resolution(kb, query)
76 #test 1
77 #(P^Q)<=>R : (Rv\sim P)v(Rv\sim Q)^(\sim RvP)^(\sim RvQ)
78 main()
79 #test 2
80 \#(P=>Q)=>Q, (P=>P)=>R, (R=>S)=>\sim(S=>Q)
81 main()
```

Terminal



```
Enter the kb:
Rv~p Rv~Q ~RvP ~RvQ
Enter the query:
```

```
Step | Clause | Derivation
```

- 1. | Rv~p | Given.
 - 2. | Rv~Q | Given. 3. | ~RvP | Given.
- 4. | ~RvQ | Given. 5. | ~r | Negated conclusion.
- 6. | P | Resolved from Rv~p and ~RvP.
- 7. Resolved from Rv~p and ~RvQ.
- ~QvP | Resolved from Rv~Q and ~RvP. 8. |
 - $R \mid Resolved from Rv~Q and Q.$