Yashasvi.R
1BM18CS154
5th Sem, B sec

2) Given (KB): $A \Rightarrow B$ and $C \Rightarrow D$, Query: $A \vee C \Rightarrow B \vee D$.
Use resolution algorithm to solve the following problem.

```
import re

def negate(term):
    return f"~{term}" if term[0] != "~" else term[1]

def reverse(clause):
    if len(clause) > 2:
        t = split_terms(clause)
        return f"{t[1]}v{t[0]}"
    return ""

def contradiction(query, clause):
    contradictions = [f"{query}v{negate(query)}",
                      f"{negate(query)}v{query}"]
    return clause in contradictions or reverse(clause)
                in contradictions.

def resolve(kb, query):
    temp = kb.copy()
    temp += [negate(query)]
    steps = dict()
    for rule in temp
        steps[rule] = "Given."
```


Yashasvi.Ravikumar

Yashasvi.R
1BM18CS154
5th Sem, B sec

```
steps[negate(query)] = "Negated Conclusion:"
i = 0
while i < len(temp):
    n = len(temp)
    j = (i + 1) % n
    clause = []
    while j != i
        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)]
                gen = t1 + t2
                if len(gen) == 2:
                    if gen[0] != negate(gen[1]):
                        clause += [f"{gen[0]} v
                                   {gen[1]}"]
                    else
                    if contradiction(query, f"{gen[0]}
                                     v{gen[1]}"):
                        temp.append(f'{gen[0]} v {gen[1]}')
                        steps[''] = f" Resolved
{temp[i]} and {temp[j]} to
{temp[-1]}
return steps
```


Yashasvi.Ravikumar

Yashaswi.R
1BM18CS154
5th Sem, B sec

```
elif len(gen) == 1:
    clauses += [f"{gen[0]}"]
else:
    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]}
                    and to {temp[-1]},
        return steps
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
return steps

def resolution(kb, query):
    kb = kb.split("")
    steps = resolve(kb, query)
    print("\nStep\t|Clause\t|Derivation\t")
    print("-" * 30)
    i = 1
    for step in steps:
        print(f"{i}.\t|{steps}\t|{steps[step]}\t")
        i += 1
```


Yashaswi.Ravikumar

Yashasvi. R
1BM18CS154
5th Sem, B sec

```
def main()
    print ("Enter the kb:")
    kb = input()
    print ("Enter the query:")
    query = input()
    resolution (kb, query)

main()
```


Yashasvi. Ravikumar