AI lab test 2

Chandan C Bagan
1BM18CS026
CSE-5A

```
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 split_terms(rule):
    exp = '(~*[PQRS])'
    terms = re.findall(exp, rule)
    return terms.
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.'
    steps[negate(query)] = 'Negated conclusion.'
    i = 0
    while i < len(temp):
        n = len(temp)
        j = (i+1) % n
        clauses = []
```

```
while i < len(temp):
    n = len(temp)
    j = (i+1) % n
    clauses = []
    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 terms if t != c]
                t2 = [t for t in terms if t != negate(c)]
                gen = t1+t2
                if len(gen) == 2:
                    if gen[0] != negate(gen[1]):
                        clauses += [f'{gen[0]}v{gen[1]}']
                    else:
                        if contradiction(query, f'{gen[0]}v{gen
                            temp.append(f'{gen[0]}v{gen[1]}
                            steps[''] = f"Resolved {temp[i]}
                            and {temp[j]} to {temp[-1]},
                            which is in turn null.\nA
                            contradiction is found when
                            {negate(query)} is assumed as tru
                            Hence, {query} is true"
                            return steps
                elif len(gen) == 1:
                    clauses += [f'{gen[0]}']

        [illegible]
```

```
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]}
        to {temp[-1]}, which is in turn null.\
        contradiction is found when {negate(query)}
        is assumed as true. Hence, {query} is
        true."
        return steps.
for clause in clauses:
    if clause not in temp and clause != reverse(clause)
    and reverse(clause) not in temp:
        Steps[clause] = f'Resolved from {temp[i]} and
                                     {temp[j]}'
        j = (j+1) % n
    i = i+1
return steps.
def resolution(kb, query):
    kb = kb.split(' ')
    steps = resolve(kb, query)
    print('\n step\t| clause\t| Deprivation\t')
    print('-' * 30)
    i = 1
    for step in steps:
        print(f' {i}. \t| {step}\t| {steps[step]}\t')
        i += 1
def main():
    print(' Enter the kb:")
    kb = input()
    print(' Enter the query:')
    query = input()
    resolution(kb, query)
```