Harsh Shankar Rao
IBM18CS032
29-12-20

Lab Test-2

Program - 3

```
def resolve(clauses, query):
    temp = clauses.copy()
    temp = [negate(query)]
    steps = dict()
    for clause in temp:
        steps[clause] = 'Given.'
    steps[negate(query)] = 'Negated conclusion'
    i = 0
    while i < len(temp):
        n = len(temp)
        j = (i+1) % n
        rules = []
        while j != i:
            term1 = splitterms(temp[i])
            term2 = splitterms(temp[j])
            for c in term1:
                if negate(c) in term2:
```

Harsh Shankar
IBM18CS032

```
r1 = [t for t in term1 if t[0] = c]
r2 = [t for t in term2 if t[0] = negate(c)]
gen = r1 + r2
if len(gen) == 2:
    if gen[0][1] != negate(gen[1][1]):
        rules += [f'{gen[0]} v {gen[1]}']
    else:
        if contradiction(goal, 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]}, which is in turn null. \nA
                contradiction is found when {negate(goal)}
            is assumed as true. Hence, {goal} is true"
        return steps
elif len(gen) == 1
    rules += [f'{gen[0]}']
else:
    if contradiction(goal, f'{term1[0]} v {term2[0]}'):
        temp.append(f'{term1[0]} v {term2[0]}')
        steps[''] = f"Resolved {temp[i]} and {temp[j]} to
{temp[-1]}, which is in turn null. \nA contradiction is found when {negate(goal)}
is assumed as true. Hence {goal} is true."
```

Harsh Shankar
IBM18CS032

return steps

~~for clause in clauses:~~

~~if rule in rules:~~

for rule in rules:

if rule not in temp and clause.rule.b = reverse(rule) and reverse(rule) not in temp:

~~temp.append(clause)~~

~~steps[clause~~

temp.append(rule)

steps[rule] = f'Resolved from {temp[i]} and {temp[j]}.'

j = (j + 1) % n

i += 1

return steps