R.K.Viswesh

241801319

DATE: 16-05-25

IMPLEMENTATION OF UNIFICATION AND RESOLUTION ALGORITHM

Program:

```
def unify(x, y, theta={}):
if theta is None:
return None     elif x ==
y:
        return theta     elif isinstance(x,
str) and x.islower():
        return unify_var(x, y, theta)     elif
isinstance(y, str) and y.islower():
        return unify_var(y, x, theta)     elif isinstance(x, list) and
isinstance(y, list) and len(x) == len(y):
        return unify(x[1:], y[1:], unify(x[0], y[0], theta))
else:
        return None


def unify_var(var, x, theta):
if var in theta:
```

```
        return unify(theta[var], x, theta)
    elif x in theta:
        return unify(var, theta[x], theta)
else:
        theta[var] = x
return theta

# Function to apply a very basic inference rule (Modus Ponens-like)
def resolution(kb, query):     for clause in kb:
        premise, conclusion = clause
theta = unify(premise, query, {})
if theta is not None:             if
conclusion == query:
                return True
return False

# Knowledge base: Human(John) → Mortal(John)
knowledge_base = [
    [["Human", "John"], ["Mortal", "John"]],
]

# Fact: Human(John) facts
= [["Human", "John"]] #
```

```
Query: Mortal(John)?
query = ["Mortal", "John"]

# Try to infer the query from the facts and knowledge base
def infer(kb, facts, query):     for fact in facts:         for rule
in kb:
            premise, conclusion = rule
theta = unify(premise, fact, {})
            if theta is not None and unify(conclusion, query, theta) is not
None:
                return True
return False

if infer(knowledge_base, facts, query):
   print("Query is resolved: John is Mortal") else:
   print("Query could not be resolved")
```

OUTPUT: