Program:

```
import re
# Function to check if two predicates can be unified
def unify(x, y, theta={}):
    if theta is None:
        return None
    elif x == y:
        return theta
    elif isinstance(x, str) and x.islower():  # x is a variable
        return unify_var(x, y, theta)
    elif isinstance(y, str) and y.islower():  # y is a variable
        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
# Function to unify a variable with a term
```

```
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 resolution rule
def resolution(kb, query):
    for clause in kb:
        theta = unify(clause[0], query, {})
        if theta is not None:
            new_kb = clause[1:]
            if not new_kb:  # If empty, means query is resolved
                return True
            else:
                return resolution(kb, new_kb[0])
    return False
# Knowledge base (Implications)
knowledge_base = [
    [["Human", "John"], ["Mortal", "John"]],  # Human(John) → Mortal(John)
]
# Fact: Human(John)
fact = ["Human", "John"]

# Query: Mortal(John)?
query = ["Mortal", "John"]
# Apply resolution
if resolution(knowledge_base, query):
    print("Query is resolved: John is Mortal")
else:
    print("Query could not be resolved")
```

Output:

```
Query is resolved: John is Mortal
```

erence between the

Result:

Thus the given case-based discussion program has been implemented successfully and the program has been uploaded in the GitHub link.