Ex No: 2b

IMPLEMENTATION OF UNIFICATION AND RESOLUTION ALGORITHM

Aim:

To implement unification and resolution algorithm using python.

Scenario:

In an AI-based expert system for **automated reasoning**, the system needs to resolve queries by **unifying logical predicates** and applying **resolution inference**. For example, given the knowledge base:

- **Rule 1:** If `John` is a **human**, then `John` is a **mortal** → `Human(John) → Mortal(John)`
- **Fact 1:** `Human(John)`
- **Query:** Is `John` mortal?

Procedure:

1. **Define the unification function (`unify`):**

- If both terms are identical, return the current substitution (`theta`).
- If one term is a **variable**, unify it with the other term.
- If both terms are **compound expressions**, unify their corresponding parts recursively.
- Otherwise, return `None` (unification fails).

2. **Define the variable unification function (`unify_var`):**

- If the variable already exists in the substitution set, apply unification recursively.
- Otherwise, assign the variable to the given term.

3. **Define the resolution function (`resolution`):**

- Iterate through the **knowledge base** (KB).
- Try to **unify** the given query with KB clauses.
- If unification succeeds, remove matched parts from KB and **recurse with the remaining** parts.
- If the knowledge base is empty after resolution, **the query is proven**.
- Otherwise, return **False** (query not proven).

4. **Provide a knowledge base with facts and implications.**

5. **Define a query to resolve (e.g., Mortal(John)).**

6. **Run the resolution function to check if the query can be proven.**

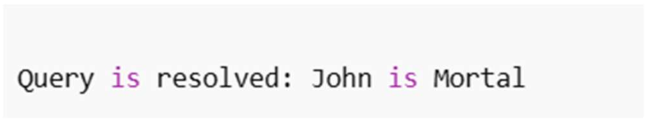
7. **Print whether the query is resolved.**

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
```