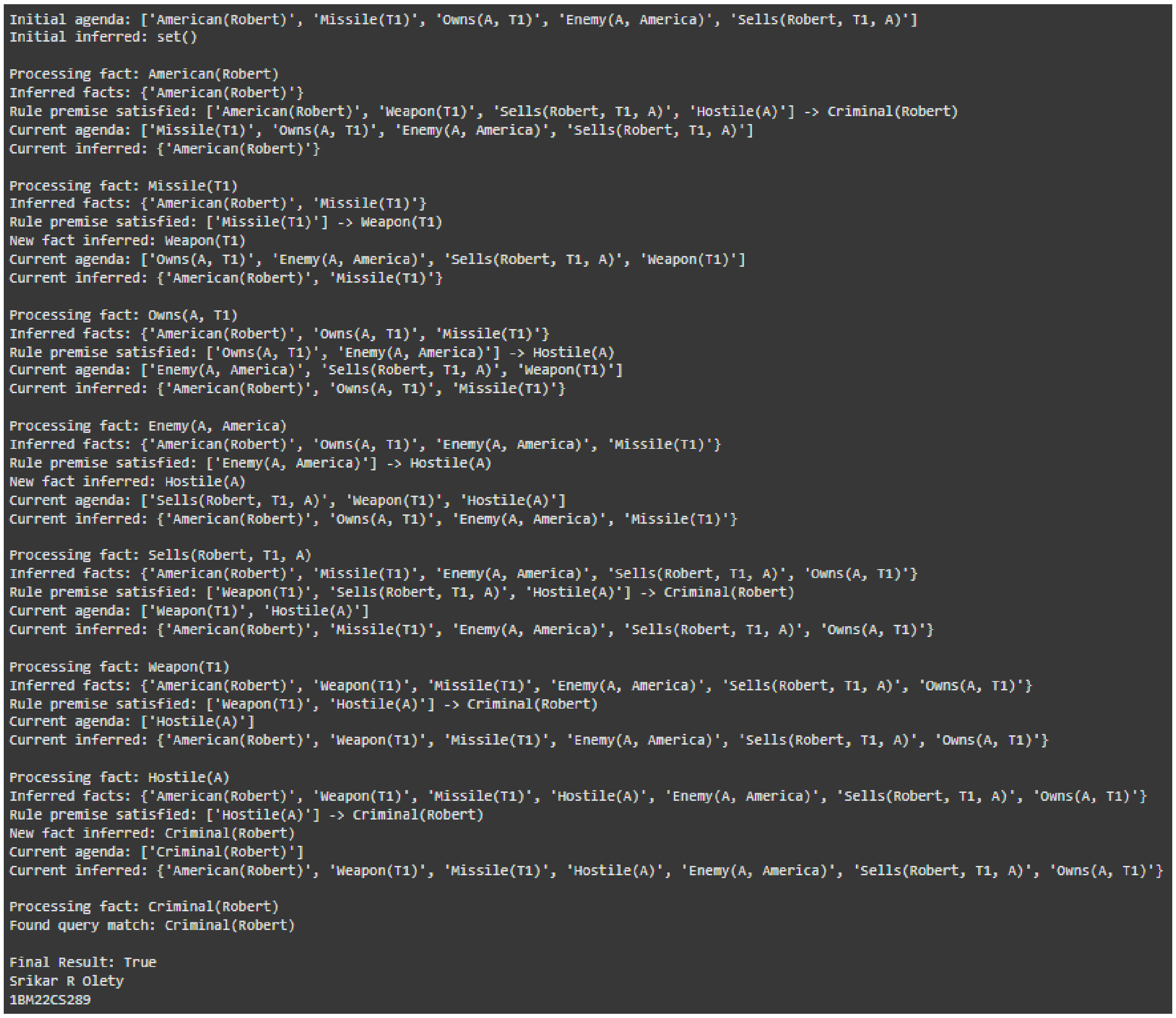
FORWARD CHAINING ALGORITHM

```python
def fol_fc_ask(KB, query):
    """
    Implements the Forward Chaining algorithm.
    :param KB: The knowledge base, a list of first-order definite clauses.
    :param query: The query, an atomic sentence.
    :return: True if the query can be proven, otherwise False.
    """
    inferred = set()  # Keep track of inferred facts
    agenda = [fact for fact in KB if not fact.get('premises')]  # Initial
facts
    rules = [rule for rule in KB if rule.get('premises')]  # Rules with
premises

    # Debugging output: Initial agenda and inferred facts
    print(f"Initial agenda: {[fact['conclusion'] for fact in agenda]}")
    print(f"Initial inferred: {inferred}")

    while agenda:
        fact = agenda.pop(0)
        print(f"\nProcessing fact: {fact['conclusion']}")

        # Check if this fact matches the query
        if fact['conclusion'] == query:
            print(f"Found query match: {fact['conclusion']}")
            return True

        # Infer new facts if this fact hasn't been inferred before
        if fact['conclusion'] not in inferred:
            inferred.add(fact['conclusion'])
            print(f"Inferred facts: {inferred}")

            # Process rules that match this fact as a premise
            for rule in rules:
                if fact['conclusion'] in rule['premises']:
                    print(f"Rule premise satisfied: {rule['premises']} ->
{rule['conclusion']}")
                    rule['premises'].remove(fact['conclusion'])  # Remove
satisfied premise
                    if not rule['premises']:  # All premises satisfied
                        new_fact = {'conclusion': rule['conclusion']}
                        agenda.append(new_fact)  # Add new fact to agenda
                        print(f"New fact inferred:
{new_fact['conclusion']}")

        # Debugging output after each iteration
        print(f"Current agenda: {[fact['conclusion'] for fact in
agenda]}")
        print(f"Current inferred: {inferred}")

    # If the loop finishes without finding the query
    print(f"Query {query} not found.")
    return False


# Example Knowledge Base
KB = [
    {'premises': [], 'conclusion': 'American(Robert)'},
    {'premises': [], 'conclusion': 'Missile(T1)'},
    {'premises': [], 'conclusion': 'Owns(A, T1)'},
    {'premises': [], 'conclusion': 'Enemy(A, America)'},
    {'premises': ['Missile(T1)'], 'conclusion': 'Weapon(T1)'},
    {'premises': ['American(Robert)', 'Weapon(T1)', 'Sells(Robert, T1,
A)', 'Hostile(A)'], 'conclusion': 'Criminal(Robert)'},
    {'premises': ['Owns(A, T1)', 'Enemy(A, America)'], 'conclusion':
'Hostile(A)'},
    {'premises': [], 'conclusion': 'Sells(Robert, T1, A)'}
]

# Query
query = 'Criminal(Robert)'

# Run the algorithm
result = fol_fc_ask(KB, query)
print("\nFinal Result:", result)
print('Srikar R Olety')
print("1BM22CS289")
```

OUPUT:



```
Initial agenda: ['American(Robert)', 'Missile(T1)', 'Owns(A, T1)', 'Enemy(A, America)', 'Sells(Robert, T1, A)']
Initial inferred: set()

Processing fact: American(Robert)
Inferred facts: {'American(Robert)'}
Rule premise satisfied: ['American(Robert)', 'Weapon(T1)', 'Sells(Robert, T1, A)', 'Hostile(A)'] -> Criminal(Robert)
Current agenda: ['Missile(T1)', 'Owns(A, T1)', 'Enemy(A, America)', 'Sells(Robert, T1, A)']
Current inferred: {'American(Robert)'}

Processing fact: Missile(T1)
Inferred facts: {'American(Robert)', 'Missile(T1)'}
Rule premise satisfied: ['Missile(T1)'] -> Weapon(T1)
New fact inferred: Weapon(T1)
Current agenda: ['Owns(A, T1)', 'Enemy(A, America)', 'Sells(Robert, T1, A)', 'Weapon(T1)']
Current inferred: {'American(Robert)', 'Missile(T1)'}

Processing fact: Owns(A, T1)
Inferred facts: {'American(Robert)', 'Owns(A, T1)', 'Missile(T1)'}
Rule premise satisfied: ['Owns(A, T1)', 'Enemy(A, America)'] -> Hostile(A)
Current agenda: ['Enemy(A, America)', 'Sells(Robert, T1, A)', 'Weapon(T1)']
Current inferred: {'American(Robert)', 'Owns(A, T1)', 'Missile(T1)'}

Processing fact: Enemy(A, America)
Inferred facts: {'American(Robert)', 'Owns(A, T1)', 'Enemy(A, America)', 'Missile(T1)'}
Rule premise satisfied: ['Enemy(A, America)'] -> Hostile(A)
New fact inferred: Hostile(A)
Current agenda: ['Sells(Robert, T1, A)', 'Weapon(T1)', 'Hostile(A)']
Current inferred: {'American(Robert)', 'Owns(A, T1)', 'Enemy(A, America)', 'Missile(T1)'}

Processing fact: Sells(Robert, T1, A)
Inferred facts: {'American(Robert)', 'Missile(T1)', 'Enemy(A, America)', 'Sells(Robert, T1, A)', 'Owns(A, T1)'}
Rule premise satisfied: ['Weapon(T1)', 'Sells(Robert, T1, A)', 'Hostile(A)'] -> Criminal(Robert)
Current agenda: ['Weapon(T1)', 'Hostile(A)']
Current inferred: {'American(Robert)', 'Missile(T1)', 'Enemy(A, America)', 'Sells(Robert, T1, A)', 'Owns(A, T1)'}

Processing fact: Weapon(T1)
Inferred facts: {'American(Robert)', 'Weapon(T1)', 'Missile(T1)', 'Enemy(A, America)', 'Sells(Robert, T1, A)', 'Owns(A, T1)'}
Rule premise satisfied: ['Weapon(T1)', 'Hostile(A)'] -> Criminal(Robert)
Current agenda: ['Hostile(A)']
Current inferred: {'American(Robert)', 'Weapon(T1)', 'Missile(T1)', 'Enemy(A, America)', 'Sells(Robert, T1, A)', 'Owns(A, T1)'}

Processing fact: Hostile(A)
Inferred facts: {'American(Robert)', 'Weapon(T1)', 'Missile(T1)', 'Hostile(A)', 'Enemy(A, America)', 'Sells(Robert, T1, A)', 'Owns(A, T1)'}
Rule premise satisfied: ['Hostile(A)'] -> Criminal(Robert)
New fact inferred: Criminal(Robert)
Current agenda: ['Criminal(Robert)']
Current inferred: {'American(Robert)', 'Weapon(T1)', 'Missile(T1)', 'Hostile(A)', 'Enemy(A, America)', 'Sells(Robert, T1, A)', 'Owns(A, T1)'}

Processing fact: Criminal(Robert)
Found query match: Criminal(Robert)

Final Result: True
Srikar R Olety
1BM22CS289
```