Forward Reasoning:-

```
class KnowledgeBase:
    def __init__(self):
        # Store facts and rules
        self.facts = set()  # Set of known facts
        self.rules = []     # List of rules (condition -> conclusion)

    def add_fact(self, fact):
        """ Add a fact to the knowledge base """
        self.facts.add(fact)

    def add_rule(self, condition, conclusion):
        """ Add a rule to the knowledge base. Condition is a callable that checks if a rule is applicable.
"""
        self.rules.append((condition, conclusion))

    def forward_reasoning(self):
        """ Perform forward reasoning to derive new facts """
        new_facts = set(self.facts)
        while True:
            added = False
            for condition, conclusion in self.rules:
                if condition(self.facts):  # If the condition is met based on current facts
                    if conclusion not in self.facts:  # If conclusion is not already a fact
                        self.facts.add(conclusion)
                        new_facts.add(conclusion)
                        added = True
            if not added:
                break  # No new facts added, stop the reasoning
        return new_facts


def get_input():
    """ Function to get user input for facts and rules """
    kb = KnowledgeBase()

    print("Enter facts (type 'done' to finish):")
    while True:
        fact = input("Fact: ").strip()
        if fact.lower() == 'done':
            break
        kb.add_fact(fact)

    print("\nEnter rules (condition -> conclusion, type 'done' to finish):")
    while True:
        rule_input = input("Rule: ").strip()
        if rule_input.lower() == 'done':
            break

        # Example rule format: "IsHuman(John) -> HasLegs(John)"
        if '->' in rule_input:
            condition, conclusion = rule_input.split('->')
            condition = condition.strip()
            conclusion = conclusion.strip()

            # Add a rule as a lambda function for condition -> conclusion
            kb.add_rule(lambda facts: condition in facts, conclusion)
        else:
            print("Invalid rule format. Please enter in the form: condition -> conclusion")

    return kb


# Main interactive loop
def main():
    print("Welcome to the Forward Reasoning System!\n")
    kb = get_input()

    # Perform forward reasoning to derive new facts
    kb.forward_reasoning()

    print("\nAll derived facts:")
    for fact in kb.facts:
        print(fact)

    # Ask the user for a query
    while True:

        query = input("\nEnter a query to check if it's a fact (e.g., HasLegs(John)): ").strip()
        if query == query.lower()=="done":
            break
        if query in kb.facts:
            print(f"Yes, {query} is a fact.")
        else:
            print(f"No, {query} is not a fact.")


if __name__ == "__main__":
    main()
```

OUTPUT

```
Welcome to the Forward Reasoning System!

Enter facts (type 'done' to finish):
Fact: IsHuman(John)
Fact: HasEyes(John)
Fact: IsHuman(Mary)
Fact: HasLegs(Mary)
Fact: DONE

Enter rules (condition -> conclusion, type 'done' to finish):
Rule: IsHuman(John) -> HasLegs(John)
Rule: IsHuman(Mary) -> HasEyes(Mary)
Rule: IsHuman(Peter) -> CanSpeak(Peter)
Rule: IsHuman(John) -> CanSpeak(John)
Rule: IsHuman(Mary) -> CanSpeak(Mary)
Rule: don
Invalid rule format. Please enter in the form: condition -> conclusion
Rule: done

All derived facts:
HasLegs(John)
IsHuman(John)
HasEyes(John)
CanSpeak(John)
CanSpeak(Mary)
CanSpeak(Peter)
HasLegs(Mary)
IsHuman(Mary)
HasEyes(Mary)

Enter a query to check if it's a fact (e.g., HasLegs(John)): HasLegs(John)
Yes, HasLegs(John) is a fact.
```