* First order logic :— DATE: 19/11/24 PAGE: 30

* Statement :- Student have passed the exam, if every student who passed the exam receives the certificate, if John is studying & passed exam, will receive a certificate.

∀ x (student(x) → passed (x, Exam))
∀ x (passed (x, Exam) → Receive certificate
student (John)
passed (John, Exam)
Receives certificate (John)

Proof ∀ x (student(x) → passed (x, Exam))
tells us that if someone is student they passed the exam.

From student (John) → passed (John, Exam) so John has passed the exam

∀ x (passed (x, Exam) → Receives certificate (x)
tells that if student has passed the exam they receive certificate.

From passed (John, Exam) ∀ → Recieves certificate)

Ans

=> Receives certificate (John) is true.

26/11/24

Lab 08

Fol

code:-

```
import re

# Define a simple function for extracting predicates from sentences
def extract_predicate(sentence):
    # Regular expression to find patterns like Predicate(Argument)
    pattern = r"([A-Za-z]+)\((\w+)\)"
    match = re.search(pattern, sentence)

    if match:
        predicate = match.group(1)
        subject = match.group(2)
        return predicate, subject
    return None, None

# Function for unification
def unify(fact, query):
    # Check if the fact and query are the same
    if fact == query:
        return True
    # Extract predicate and subject from fact and query
    fact_predicate, fact_subject = extract_predicate(fact)
    query_predicate, query_subject = extract_predicate(query)
```

```
    # If predicates match, unify the subjects
    if fact_predicate == query_predicate:
        if fact_subject == query_subject:
            return True
        else:
            # Here, we could handle variable substitution (unification)
            return False
    return False

# Function to deduce the goal using given rules
def deduct(rules, goal):
    # Try to find unification for the goal from the rules
    for rule in rules:
        if unify(rule, goal):
            print(f"Unification successful: {rule} matches with {goal}.")
            return True
    return False

# Main function to handle user input
def main():
    # Step 1: Get the rules (facts/implications) from the user
    print("Enter the rules (facts/implications). Type 'done' to finish entering rules.")
    rules = []
```

```
    while True:
        rule_input = input("Enter rule: ")
        if rule_input.lower() == 'done':
            break
        else:
            rules.append(rule_input.strip())

    # Step 2: Get the goal (query) from the user
    goal_input = input("Enter the goal (query) to prove: ").strip()

    # Step 3: Try to deduce the goal using the given rules
    print("\nAttempting to deduce the goal...")
    if deduct(rules, goal_input):
        print(f"Conclusion: The goal '{goal_input}' is true based on the rules.")
    else:
        print(f"Conclusion: The goal '{goal_input}' cannot be proven with the
provided rules.")

# Run the program
main()
```

output:

```
Enter the rules (facts/implications). Type 'done' to finish entering rules.
Enter rule:  Loves(Sam, Everyone)
Enter rule: done
Enter the goal (query) to prove: Loves(Everyone, Sam)

Attempting to deduce the goal...
Unification successful: Loves(Sam, Everyone) matches with Loves(Everyone, Sam).
Conclusion: The goal 'Loves(Everyone, Sam)' is true based on the rules.
```