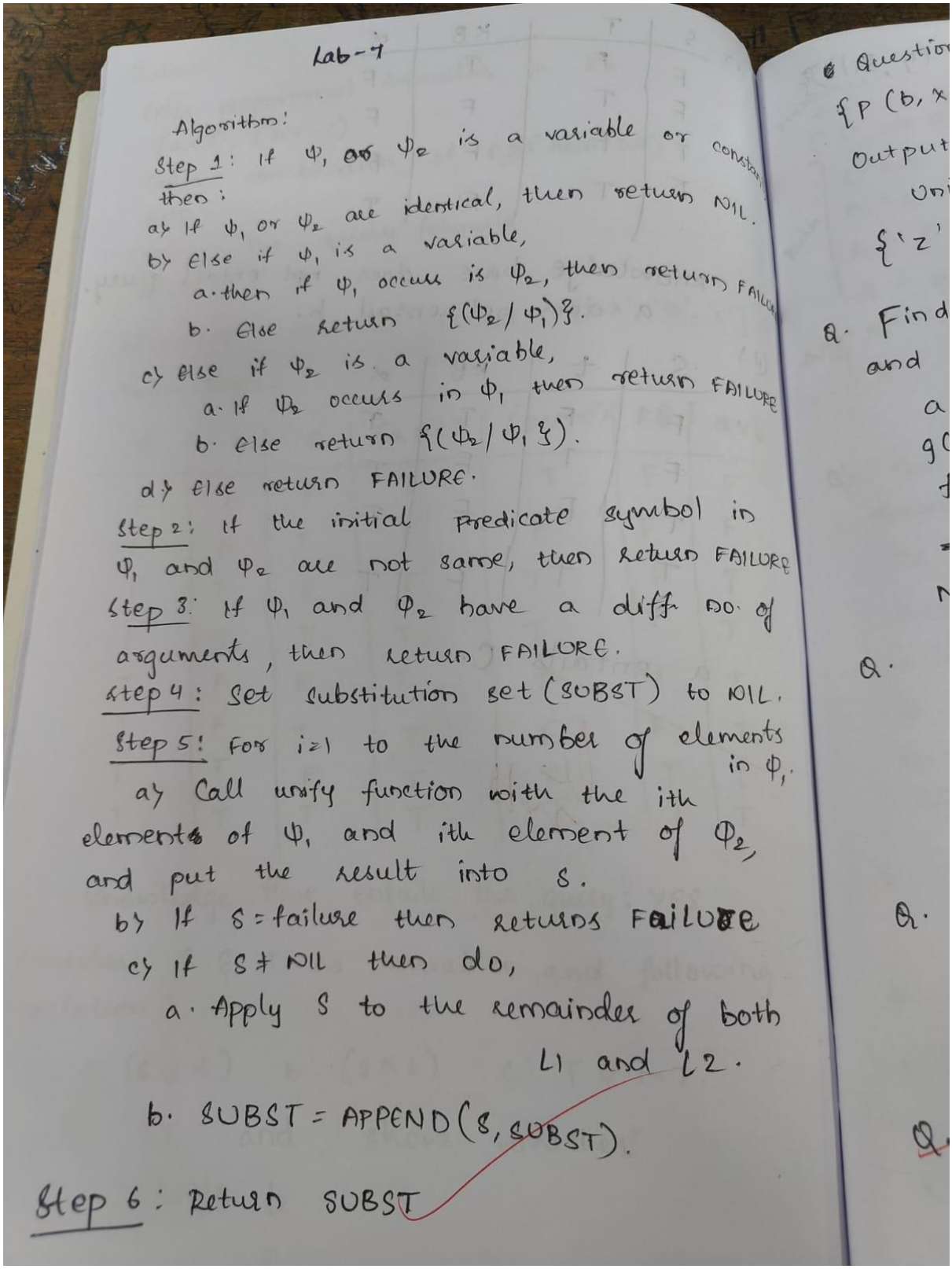
Week – 7

Implement unification in first order logic.

Algorithm:



Lab-7

Algorithm:

Step 1: If Ψ_1 or Ψ_2 is a variable or constant then:

a) If Ψ_1 or Ψ_2 are identical, then return NIL.

b) Else if Ψ_1 is a variable,
   a. then if Ψ_1 occurs is Ψ_2, then return FAILURE
   b. Else return $\{(\Psi_2 / \Psi_1)\}$.

c) Else if Ψ_2 is a variable,
   a. If Ψ_2 occurs in Ψ_1 then return FAILURE
   b. Else return $\{(\Psi_2 / \Psi_1 \}$.

d) Else return FAILURE.

Step 2: If the initial predicate symbol in Ψ_1 and Ψ_2 are not same, then return FAILURE

Step 3: If Ψ_1 and Ψ_2 have a diff no. of arguments, then return FAILURE.

Step 4: Set substitution set (SUBST) to NIL.

Step 5: For i=1 to the number of elements in Ψ_1.

a) Call unify function with the ith elements of Ψ_1 and ith element of Ψ_2, and put the result into S.

b) If S = failure then returns FAILURE

c) If S ≠ NIL then do,

a. Apply S to the remainder of both L1 and L2.

b. SUBST = APPEND(S, SUBST).

Step 6: Return SUBST

* Question:
{P(b, x, f(g(z)) and P(z, f(y), f(y))}.

Output:
Unification Result is ~~false~~.
{'z': 'b', 'x': ['f', 'y'], 'y': ['g', b]}

Q. Find MGU of {Q(a, g(x,a), f(y)) and Q(a, g(f(b), a), x)}.

a = a
g(x,a) = g(f(b), a) ⇒ x = f(b)
f(y) = x → substitue x = f(b)
→ f(y) = f(b)

⇒ y = b
MGU = {x(f(b)), y/b}.

Q. Unify for {P{f(a), g(y)), P(x, x)}.
f(a) = x and g(y) = x
⇒ f(a) = g(y)
MGU = FAIL

Q. MGU of {prime (11) and prime(y)}
11 → y → y = 11
MGU = {y/11}

Q. MGU {knows(John, x), knows(y, mother(y))}

John = y x = (mother (y))
⇒ x = mother (John)
MGU = {y/John, x/mother (John)}

Output:

```
Unification succeeded with substitution: {'x': 'B', 'y': 'A'}
```

Code:

```
def unify(x, y, subst=None):
    if subst is None:
        subst = {}

    # If x or y is a variable or constant
    if is_variable(x) or is_constant(x):
        if x == y:
            return subst
        elif is_variable(x):
            return unify_var(x, y, subst)
        elif is_variable(y):
            return unify_var(y, x, subst)
        else:
            return None

    # If both x and y are compound expressions
    if is_compound(x) and is_compound(y):
        if x[0] != y[0] or len(x[1]) != len(y[1]):
            return None
        for xi, yi in zip(x[1], y[1]):
            subst = unify(xi, yi, subst)
            if subst is None:
                return None
        return subst
    return None

def is_variable(x):
    return isinstance(x, str) and x.islower() and x.isalpha()

def is_constant(x):
```

```
    return isinstance(x, str) and x.isupper() and x.isalpha()

def is_compound(x):
    return isinstance(x, tuple) and len(x) == 2 and isinstance(x[0], str) and isinstance(x[1], list)

def unify_var(var, x, subst):
    if var in subst:
        return unify(subst[var], x, subst)
    elif x in subst:
        return unify(var, subst[x], subst)
    elif occurs_check(var, x, subst):
        return None
    else:
        subst[var] = x
        return subst
```

49

```
def occurs_check(var, x, subst):
    if var == x:
        return True
    elif is_variable(x) and x in subst:
        return occurs_check(var, subst[x], subst)
    elif is_compound(x):
        return any(occurs_check(var, arg, subst) for arg in x[1])
    else:
        return False

# Example usage:
# Let's say we want to unify P(x, A) and P(B, y)
x = ("P", ["x", "A"])
y = ("P", ["B", "y"])
```

```
result = unify(x, y)
if result is not None:
    print("Unification succeeded with substitution:", result)
else:
    print("Unification failed.")
```