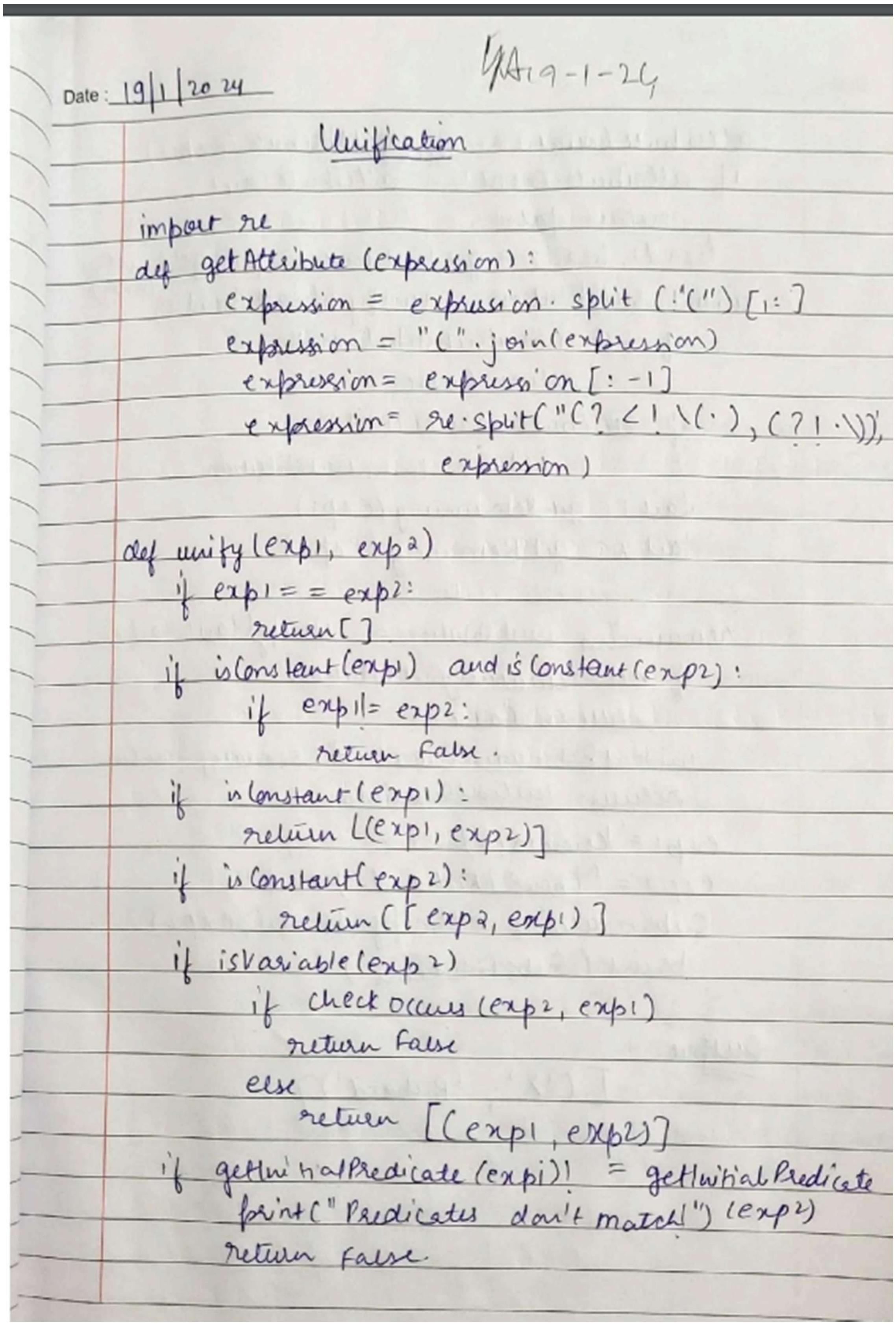
Program 8 : Unification

Code:

```
import re

def getAttributes(expression):
    expression = expression.split("(")[1:]
    expression = "(".join(expression)
    expression = expression[:-1]
    expression = re.split("(?<!\(.),(?!.\))", expression)
    return expression

def getInitialPredicate(expression):
    return expression.split("(")[0]

def isConstant(char):
    return char.isupper() and len(char) == 1

def isVariable(char):
    return char.islower() and len(char) == 1

def replaceAttributes(exp, old, new):
    attributes = getAttributes(exp)
    for index, val in enumerate(attributes):
        if val == old:
            attributes[index] = new
    predicate = getInitialPredicate(exp)
    return predicate + "(" + ",".join(attributes) + ")"

def apply(exp, substitutions):
    for substitution in substitutions:
        new, old = substitution
        exp = replaceAttributes(exp, old, new)
    return exp
def checkOccurs(var, exp):
    if exp.find(var) == -1:
        return False
    return True



def getFirstPart(expression):
    attributes = getAttributes(expression)
    return attributes[0]



def getRemainingPart(expression):
    predicate = getInitialPredicate(expression)
    attributes = getAttributes(expression)
    newExpression = predicate + "(" + ",".join(attributes[1:]) + ")"
    return newExpression
def unify(exp1, exp2):
    if exp1 == exp2:
        return []

    if isConstant(exp1) and isConstant(exp2):
        if exp1 != exp2:
            return False

    if isConstant(exp1):
        return [(exp1, exp2)]

    if isConstant(exp2):
        return [(exp2, exp1)]

    if isVariable(exp1):
        if checkOccurs(exp1, exp2):
            return False
        else:
            return [(exp2, exp1)]

    if isVariable(exp2):
        if checkOccurs(exp2, exp1):
            return False
        else:
            return [(exp1, exp2)]

    if getInitialPredicate(exp1) != getInitialPredicate(exp2):
        print("Predicates do not match. Cannot be unified")
        return False

    attributeCount1 = len(getAttributes(exp1))
    attributeCount2 = len(getAttributes(exp2))
    if attributeCount1 != attributeCount2:
        return False

    head1 = getFirstPart(exp1)
    head2 = getFirstPart(exp2)
    initialSubstitution = unify(head1, head2)
    if not initialSubstitution:
        return False
    if attributeCount1 == 1:
        return initialSubstitution
    tail1 = getRemainingPart(exp1)
    tail2 = getRemainingPart(exp2)

    if initialSubstitution != []:
        tail1 = apply(tail1, initialSubstitution)
        tail2 = apply(tail2, initialSubstitution)

    remainingSubstitution = unify(tail1, tail2)
    if not remainingSubstitution:
        return False

    initialSubstitution.extend(remainingSubstitution)
    return initialSubstitution
exp1 = "knows(X)"
exp2 = "knows(Richard)"
substitutions = unify(exp1, exp2)
print("Kanjika Singh-1BM21CS086")
print("Substitutions:")
print(substitutions)
exp1 = "knows(A,x)"
exp2 = "knows(y,mother(y))"
substitutions = unify(exp1, exp2)
print("Substitutions:")
print(substitutions)
```

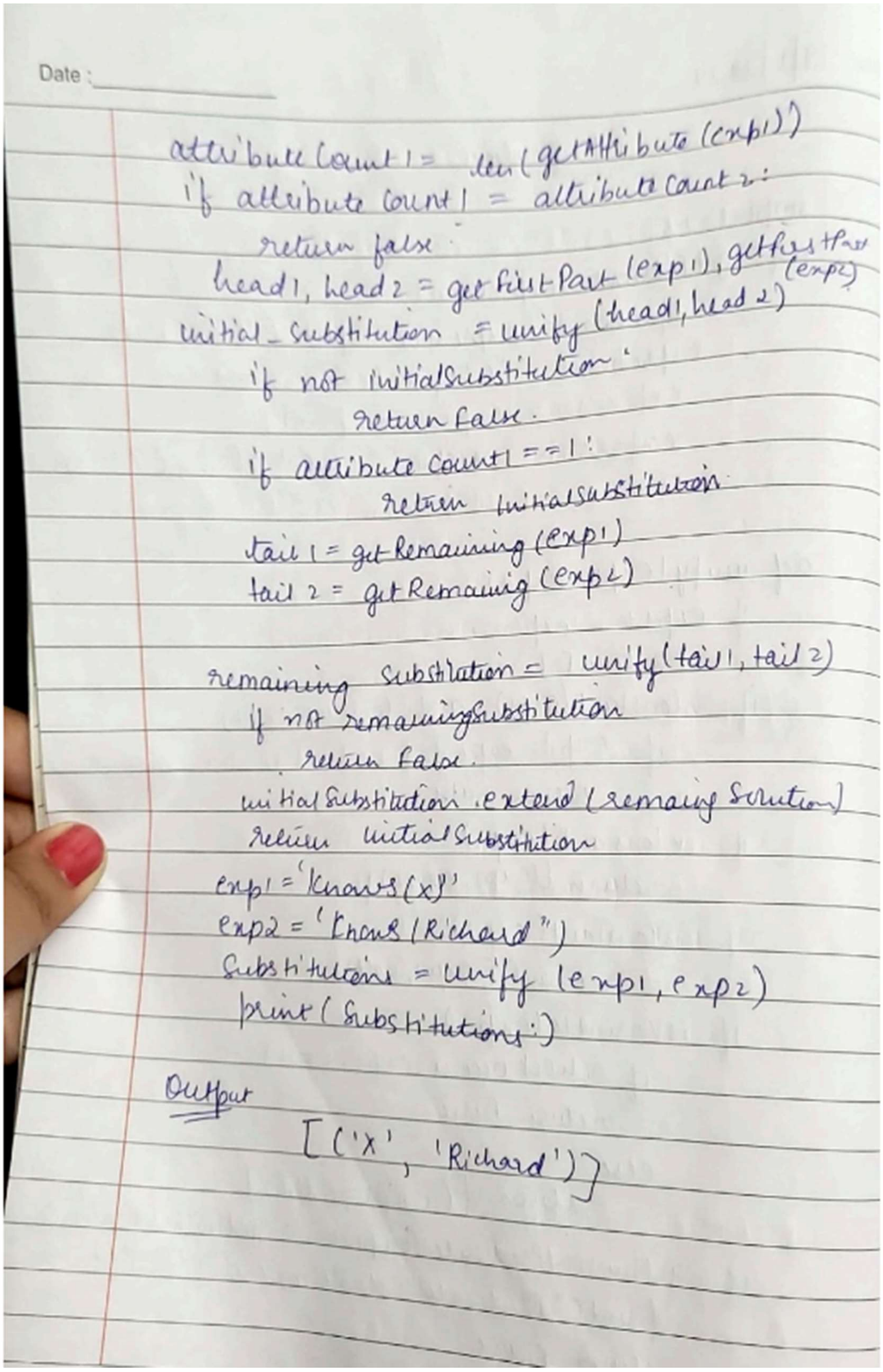
Observation :



Date : 19/1/2024 [illegible] 19-1-24

Unification

```
import re
def getAttribute (expression):
        expression = expression.split ("(") [1:]
        expression = "(".join(expression)
        expression= expression [: -1]
        expression= re.split("(?<!\(.), (?!.\))",
                        expression )

def unify(exp1, exp2)
    if exp1 = = exp2:
        return[]
    if isConstant(exp1)  and isConstant(exp2):
        if exp1!= exp2:
            return False.
    if isConstant(exp1):
        return [(exp1, exp2)]
    if isConstant(exp2):
        return ([exp2, exp1) ]
    if isVariable(exp2)
        if checkOccurs(exp2, exp1)
            return False
        else
            return [(exp1, exp2)]
    if getInitialPredicate(exp1)! = getInitialPredicate (exp2)
        print("Predicates don't match!")
        return False.
```



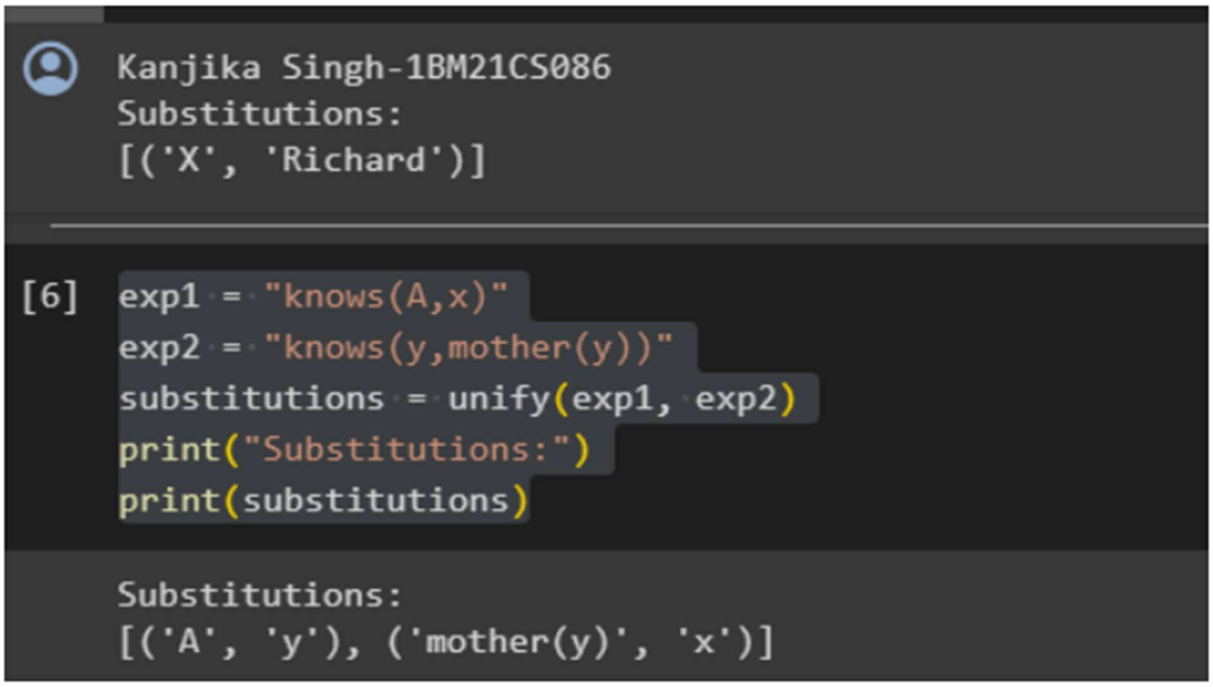
Date :

```
attributeCount1 = len(getAttribute(exp1))
 if attribute count1 = attribute count 2:
     return false
   head1, head2 = get first Part (exp1), getFirstPart
                                                 (exp2)
 initial_substitution = unify (head1, head2)
     if not initialSubstitution:
           return false.
     if attribute count1 == 1:
               return initialSubstitution
     tail 1 = getRemaining (exp1)
     tail 2 = getRemaing (exp2)

   remaining substitution = unify(tail1, tail2)
     if not remainingSubstitution
        return false.
     initialSubstitution.extend (remaing Solution)
     return initialSubstitution
   exp1 = 'knows(x)'
   exp2 = 'know (Richard")
   Substitutions = unify (exp1, exp2)
    print( Substitutions:)
```

Output

```
          [('x', 'Richard')]
```

Output:



```
Kanjika Singh-1BM21CS086
Substitutions:
[('X', 'Richard')]
```

```
[6] exp1 = "knows(A,x)"
    exp2 = "knows(y,mother(y))"
    substitutions = unify(exp1, exp2)
    print("Substitutions:")
    print(substitutions)
```

```
Substitutions:
[('A', 'y'), ('mother(y)', 'x')]
```