Program 8: Unification

```
Code:
import re
def getAttributes(expression):
    expression = expression.split("(")[1:]
    expression = "(".join(expression)
    expression = expression[:-1]
    expression = re.split("(?<!\(.),(?!.\))", expression)</pre>
    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 MAI9-1-24
	Unification
	a death and a second of the second
	import re
	des get Attribute (expression):
	expression = expression split (:'(") [:]
	expansion = "(".join(expression)
	expression = expression[:-1]
	expression= 2e.spit("(? (()), (?1.)))</td
	expression)
	Cold St. State Land and Astronomy All Edition
	def unity (exp1, exp2)
	exp = exp2:
	return []
	if is (onstant (exp)) and is constant (expr):
	if expl= exp2:
	return False.
	if in (onstant (exp1):
	return L(exp1, exp2)]
	if is Constant (exp2):
	return (texpa, empi)]
To a second	if is Variable (exp 2)
	if check occurs (exp2, exp1)
	neturn false
	else
	return [(exp1, exp2)]
	if gettin halfredicate (expi)! = gettinhal Predicate
_	frint ("Predicates don't match!") (exp2)
	return faire

	attribute (count 1 = den (getAttribute (coups))
	if alleibute count 1 = altribute count 2:
	return false
	head, head = get first Part (exp1) of leaper
	head, head 2 = get first Part lexp1), getfirstforence this transfer (thead, head 2)
	if not initialsubstitution.
	netuen fall.
	1
	netren furasion
	F
	tail 2 = get Remaining (expl)
	white (tail tail 2)
	remaining substitution = unify(tail, tail2)
	If not remaininguiss with
	mitial Substitution extend (remains Scrution)
	Relieu litial Substitution
	enp1 = 'Knows(x)'
1	enp2 = 'knows (Richard")
1	substitutions = living leads e a
	print (Substitutions:)
	Output
	[('x', 'Richard')]
) marada)

Output:

```
(2) 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')]
```