Program -8
Unification

```
import re

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

def unify(exp1, exp2):
    if exp1 == exp2:
        return []
    if is_constant(exp1) and is_constant(exp2):
        if exp1 != exp2:
            return False
    if is_variable(exp1):
        if check_occurs(exp1, exp2):
            return False
        else
            return [(exp2, exp1)]
    if is_variable(exp2):
        if check_occurs(exp1, exp2):
            return False
        else
            return [(exp2, exp1)]
    if getInitialPredicate(exp1) != getInitialPredicate(exp2):
        print("Predicates do not match. Cannot be unified")
        return False
    head1 = getFirst
    head2 = 6 " (exp2)
    initialSubstitution = unify(head1, head2)
    if not initialSubstitution:
        return False
    if attributeCount(exp1) == 1:
        return initialSubstitution
    tail1 = apply(tail1, initialSubstitution)
    tail2 = " " 2
    remainingSolution = unify(tail1, tail2)
    if not remainingSolution:
        return False
    exp1 = knows(X)
    exp2 = " (Richard)
    Substitution = unify(exp1, exp2)
    print("Substitution")
```

Output: [('X', 'Richard')]

```
exp1 = knows(A,x)
exp2 = knows(y, mother(y))
```

Substitution:

[(A, y), (mother(y), x)]

Algorithm

function works by breaking down exp into their components, handling constants, variables and predicates appropriately, and recursively verifying the substitutions

→ If the 2 expressions are identical, returns an empty list (No Subs needed)

→ If 1 of the expression is a constant, unifies by substituting constant for the variable

→ If one of the expression is a variable, unifies by substituting variable for the constant

→ If the predicates of the exps don't match, unification fails

→ Recursively performs unification on the head & tail of the expressions

```
[ ]  exp1 = "knows(X)"
     exp2 = "knows(Richard)"
     substitutions = unify(exp1, exp2)
     print("Substitutions:")
     print(substitutions)
```

```
     Substitutions:
     [('X', 'Richard')]
```

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

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