Code

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

hears(John, x) —①
hears(John, Jane) —②.

```
def unify(exp1, exp2):
    if exp1 == exp2: #Checks the entire expression
        return[]
    if isConstant(exp1) and isConstant(exp2):
        if exp1 != exp2:
            return False
    if isConstant(exp1):
        return [(exp1, exp2)]  # returns substitution list.
    if isConstant(exp2):
        return ([exp2, exp1)].
```

knows(John, x)
knows(Jane, y).

```
    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 don't match!");
        return False
```

```
attributeCount1 = len(getAttributes(exp1))
--------- 2 = ---------- 2
if attributeCount1 != attributeCount2:
    return False

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

tail1 = getRemainingPart(exp1)
Tail2 = ----------(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("Substitutions:")
print(substitutions)
```

Output:

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

```
exp1 = "knows(A, X)"
exp2 = "knows(y, mother(y))"
```

Output: Substitutions:

```
[('A', 'y'), ('mother(y)', 'X')]
```

Unification — making 2 expressions look identical.

Conditions:

- The predicate should be the same
- The no. of arguments in both the expressions must be the same.
- If a similar variables are present in the same exp, then unification fails.

$P(x, F(y))$ — ①

$P(a, F(g(z))$ — ②

$[a/x]$ (x with a)

$P(a, F(y))$, $P(a, F(g(z))$ $[g(z)/y]$

$P(a, F(g(z))$ - $P(a, F(g(z))$

```
107    exp1 = "knows(A,x)"
108    exp2 = "knows(y,Y)"
109    substitutions = unify(exp1, exp2)
110    print("Substitutions:")
111    print(substitutions)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\neha2\OneDrive\Documents\NehaKamath_1BM21CS113_AILab> python
Substitutions:
[('A', 'y'), ('Y', 'x')]
```