9) Infer whether given pair of statements can be unified or not. If unification is possible write the code for substitutions:
1) Predicates are different 2) Mismatch in no of arguments
3) If the arguments are constants.

```
import re
def getAttributes(expression):
    expression = expression.split("(")[1:]
    expression = "(".join(expression)
    expression = expression.split(")")[:-1]
    expression = ")".join(expression)
    attributes = expression.split(",")
    return attributes
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)
    predicate = getInitialPredicate(exp)
    for index, val in enumerate(attributes):
        if val==old
            attributes[index] = new
    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:
            print("{exp1} and {exp2} are
                  constant, cannot be unified")
            return []
```

```
if isConstant(exp1) and isConstant(exp2):
    if exp1 != exp2:
        print("{exp1} and {exp2}
if isConstant(exp1):
    return [(exp1, exp2)]
if isConstant(exp2):
    return [(exp2, exp1)]
if isVariable(exp1):
    return [(exp2, exp1)] if not checkOccurs
                          (exp1, exp2) else []
if isVariable(exp2):
    return [(exp1, exp2)] if not checkOccurs(exp2,
                                             exp1)
                          else []
if getInitialPredicate(exp1) != getInitialPredicate(exp2):
    print("Cannot be unified as the predicates do
          not match!")
    return []
attributeCount1 = len(getAttributes(exp1))
attributeCount2 = len(getAttributes(exp2))
if attributeCount1 != attributeCount2:
    print("Length of attributes {attributeCount1} and
          {attributeCount2} do not match. Cannot be
          unified")
return []
```

```
head1 = getFirstPart(exp1)
head2 = getFirstPart(exp2)
initialSubstitution = unify(head1, head2)
if not initialSubstitution:
    return []
if attribute count1 == 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 []
return initialSubstitution + remainingSubstitution

def main():
    print("Enter the first expression")
    e1=input()
    print("Enter the second expression")
    e2= input()
    substitutions = unify(e1, e2)
    print("The substitutions are:")
    print([' / '.join(substitution) for substitution in
                                    substitution])
```