Pooja Srinivasan
IBM18CS069
AI LAB TEST-2
Pooja Srinivasan
1-1-21

Prg - Infer whether the given pair of sentences can be unified.

```
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 InitialPredicate(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 = getPredicate(exp)
    for index, val in enumerate(attributes):
        if val == old:
            attributes[index] = new
    return predicate + "(" + ",".join(attributes) + ")"
```

Pooja Srinivasan
1BM18CS069
Pooja Srinivasa
1-1-21

```
def apply(exp, substitutions):
    for substitution in substitutions:
        new, old = substitution
        exp = replaceAttribute(exp, old, new)
    return exp
```

```
def CheckOccur(var, exp):
    if exp.find(var) == -1:
        return False
    return True
```

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

```
def getRemainingPart(expression):
    predicate = getPredicate(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(f"{exp1} and {exp2} are constants. Cannot be unified")
            return[]
```

Pooja Srinivasan
1BM18CS069
AI lab test-2
Pooja Srinivasan
1-1-21

```
if isConstant(exp1):
    return [(exp1, exp2)]
if isConst(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 getIntialPredicate(exp1) != getIntialPredicate(exp2):
        print("Cannot be unified as the predicates do not match")
        return []
attribute_count1 = len(getAttributes(exp1))
attribute_count2 = len(getAttribute(exp2))
if attribute_count1 != attribute_count2
    print(f"Length of attributes {attribute_count1} and
          {attribute_count2} do not match. Cannot be unified")
head1 = getFirstPart(exp1)
head2 = getFirstPart(exp2)
initialSubstitution = unify(head1, head2)
if not initialSubstitution:
    return []
if attributeCount1 == 1:
    return initialSubstitution
tail1: getRemainingPart(exp1)
tail2: getRemainingPart(exp2)
```

```
if initialSubstition != []
        tail1 = apply (tael 1, initialSubstion)
        tail2 = apply(tail2, initialSubstion)

    remaingSubstion = unify (tail1, tail2)
    if not remaingSubstition:
            return []

    return initialSubstion + remaingSubstion

def main():
    print ("Enter 1st expression)
    e1=input()
    print ("Enter 2nd expression)
    substitutions = unify (e1, e2)
    print ("The substitutions are:")
    print ([' / '.join [substion) for substitiontion in substitutions])
```