AI Lab-Test-2

Venkatesha Prasada CH
1BM18CS124
5th C-2

Qn: Inter whether given program can be unified or not if unification is possible write the code for Substitution. Justify your ans for "Likes(Ram, u) and Likes(x, Raj)"

Ans:

```
import re.

def getAttribute(sentence):
    sentence = sentence.split("(")[1:]
    sentence = "(".join(sentence)
    sentence = sentence[:-1]
    sentence = re.split("(?<!\(.),(?!.\))", sentence)
    return sentence

def getInitialPredicate(sentence):
    return sentence.split("(")[0]

def isVariable(char):
    return char.islower() and len(char)==1

def isConstant(char):
    return char.isupper() and len(char)==1

def replaceAttributes(sen, old, new):
    attributes = getAttributes(sen)
    for index, val in enumerate(attributes):
        if val == old:
```

```
        attributes[index] = new
    predicate = getInitialPredicate(sen)
    return predicate + "(" + "," .join(attributes) + ")"

def apply(sen, substitutions):
    for substitution in substitutions:
        new, old = substitution
        sen = replaceAttributes(sen, old, new)

    return sen

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.
```

Venkatesha Prasada CG
1BM18CS124

```
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("Cannot be unified")
        return False
    attribute count1 = len(getAttributes(exp1))
    attribute count2 = 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)
    tail12 = apply(tail2, initial substitution)

remainingSubstitution = unify(tail1, tail2)
if not remainingSubstitution:
    return False
initial substitution.extend(remainingsubstitutions)
return initial substitution
```

```
exp1 = Likes(Ram, y)
exp2 = Likes(x, Raj)
sub = unify(exp1, exp2)
print("sub", sub)
```