Anshul
Shetty
1BM18CS129

Check whether the statements can be unified, If unification is possible, write the code for substitutions

Justify your answer for "Likes(Ram, y) and Likes(x, Raj)"

↳) ans. Unification is possible.

```
def unify (expr1, expr2)
  If expr1 == expr2 :
     return []
  If isConstant(expr1) and
   isConstant(expr2) :
       if expr1 != expr2 : return False
  If isConstant(expr1):
       return [(expr2, expr1)]
  If isConstant(expr2) :
     return [(expr1, expr2)]
  If isVariable(expr1):
     If checkOccurs(expr1, expr2):
        return False
     else : return [(expr2, expr1)]
  If isVariable(expr2):
     If checkOccurs(expr2, expr1):
        return false
     else:
        return [(expr1, expr2)]
```

```
if initialPredicate(expr1) != initialPredicate(expr2):
    print("cannot be unified")
    return False

atb_count1 = len(getAttributes(expr1))
atb_count2 = len(getAttributes(expr2))
if atbcount1 != atb_count2:
    return False
head1 = getFirstPart(expr1)
head2 = getFirstPart(expr2)

initial_sub = unify(head1, head2)
if not initial_sub:
    return False
if atb_count1 == 1:
    return initial_sub

tail1 = getRemainingPart(expr1)
tail2 = getRemainingPart(expr2)

if initial_sub != []:
    tail1 = apply(tail1, initial_sub)
    tail2 = apply(tail2, initial_sub)
rem_sub = unify(tail1, tail2)
if not rem_sub:
    return False
initial_sub.extend(rem_sub)
res = []
```

```
    for tup in initial_sub:
        st = '/'.join(tup)
        res.append(st)
    return res.

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

def checkOccurs(var, expr):
    if expr.find(var) == -1:
        return False
    return True

def getFirstPart(expr):
    attr = getAttributes(expr)
    return attr[0]

def getremainingPart(expr): predicate = getInitialpredicate(expr)
    attr = getAttributes(expr)
    newExpr = predicate + "(" + ",".join
        (attr[1:] + ")"
    return newExpr

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