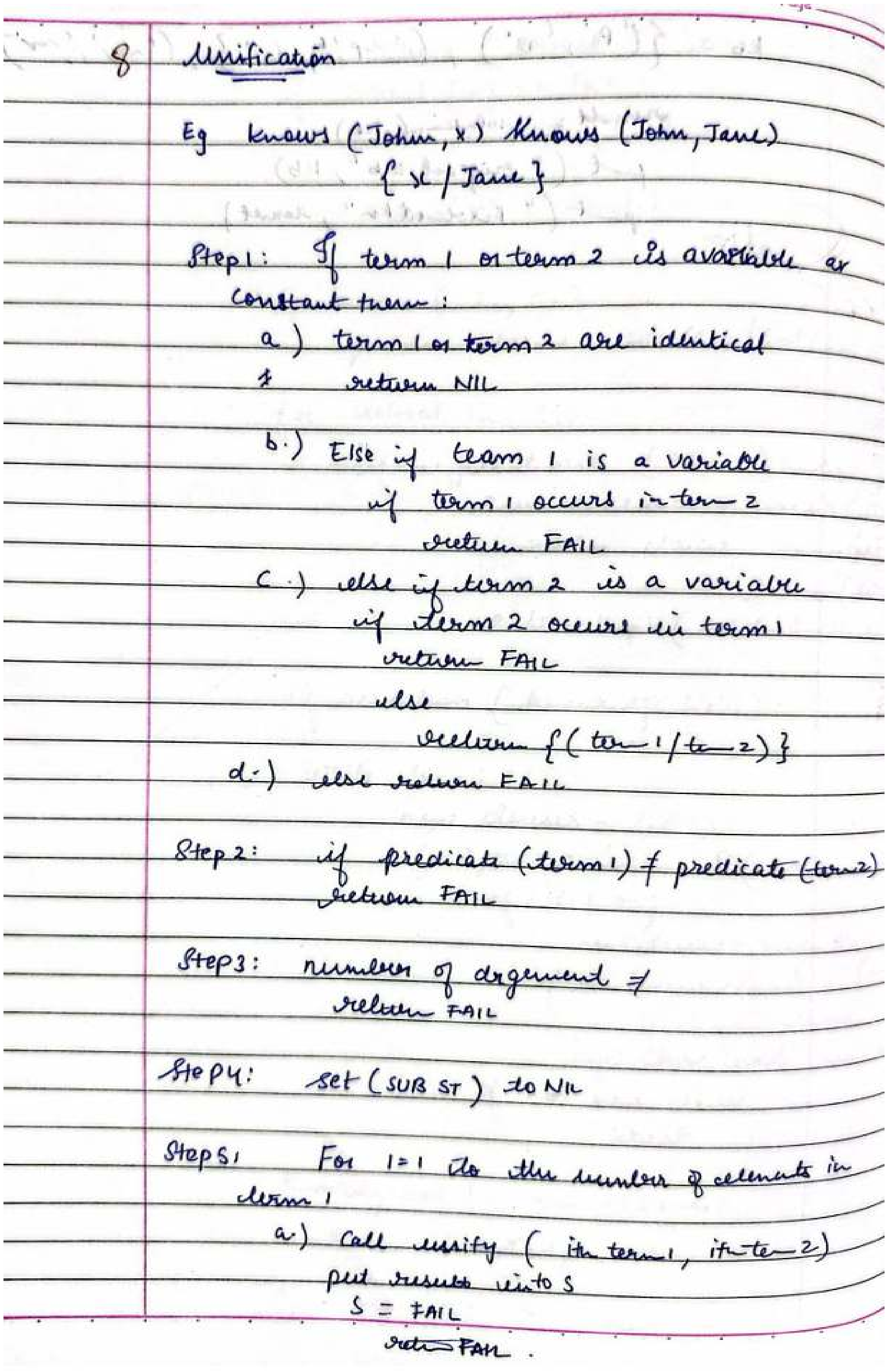
Unification
Code:-



8 Unification

Eg knows (John, x) Knows (John, Jane)
{x / Jane}

Step1: If term 1 or term 2 is available as constant then:
a) term 1 or term 2 are identical
≠ return NIL
b.) Else if team 1 is a variable
if term 1 occurs in term 2
return FAIL
c.) else if term 2 is a variable
if term 2 occurs in term 1
return FAIL
else
return {(term 1/term 2)}
d.) else return FAIL

Step2: if predicate (term 1) ≠ predicate (term 2)
return FAIL

Step3: number of argument ≠
return FAIL

Step4: set (SUB ST) to NIL

Step5: For I=1 to the number of elements in term 1
a) call unify (ith term1, ith term 2)
put results into S
S = FAIL
return FAIL.

~~Step5~~: c) if S ≠ NIL
a.) Apply S to the remainder of both L1 & L2
b.) SUBST . APPEND (S, SUBST)

Step 6: Return SUBST

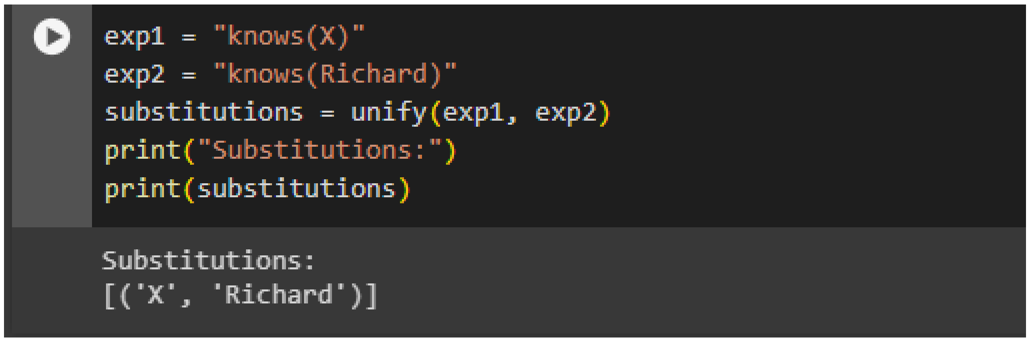
① predicate same
② No of arguments
③ John   John
X   Jane
then put in subset

[illegible] 19/11

```
import re
def getInitialPredicate (expression):
    return expression.split ("(")[0]
def isConstant (char):
        return char.isupper() and len(char)==1
def replaceAttributes (exp, old, new):
        attributes = getAttributes (exp)
        for index, val in enumerate (attributes).
            if val == old :
                attributes [index] = new
            predicate = getInitial predicate (exp)
        return predicate + "(" + ",".join (attributes) + ")"
def apply (exp, substitutions):
    for substitution in substitutions:
            new , old = substitution
            exp = replaceAttributes (exp, old, new)
        return exp.

def getFirstPart (expression):
    predicate = getInitialPredicate (expression)
    attributes = getAttributes (expression)
    return new Expression.
```

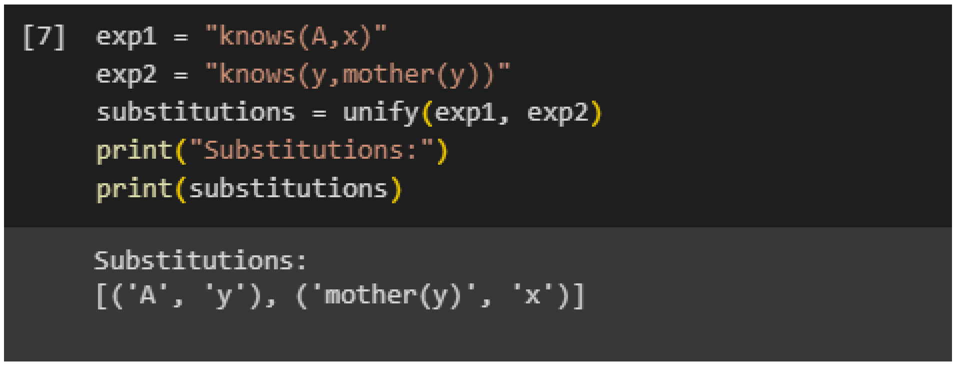
```
defunify (exp1, exp2)
        if exp1 == exp2
            return false.
        if isconstant ( exp1 ):
            return [(exp2, exp1)]
        if isVariable (exp 1):
            if checkOccurs (exp1, exp2):
                return False
   attribute count1 = len (getAttributes (exp1))
   attribute Count 2 = len (getAttributes (exp2))
       if attributeCount1 != attribute count2:
            return false.
       head1 = get FirstPart (exp1)
       head 2 = get firstPart (exp2)
       initial Substitution = unify ( head1, head2)
         if not initial. Substitution :
              return False.
              return intial Substitution

         tail 1 = getRemainPart (exp1)
         tail 2 = getRemaingPart (exp2)

      . initial Substitution . extend (remaingSubstitution)
      return initial Substitution.
```

output:-



```
exp1 = "knows(X)"
exp2 = "knows(Richard)"
substitutions = unify(exp1, exp2)
print("Substitutions:")
print(substitutions)
```

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



```
[7] exp1 = "knows(A,x)"
    exp2 = "knows(y,mother(y))"
    substitutions = unify(exp1, exp2)
    print("Substitutions:")
    print(substitutions)
```

```
Substitutions:
[('A', 'y'), ('mother(y)', 'x')]
```