1/1/2021 AI - LAB - TEST - 2

REVANTH. R

1BM18CS082

(B-2)

① Infer unification code for

Substitutions p(5, X), f(g(21) and p(Z, f(Y), f(4))

1) Predicates are diffferent 2) Mismatch in number of arguments

3) If 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
```

(1A)

@revanth

AI - LAB - TEST - 2

REVANTH.R
1BM18CS082
@vfanth

```
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
```

AI - LAB - TEST - 2

REVANTH.R
1BM18CS082

```
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):
    attributes = getAttributes(expression)
    ~~return attributes[0]~~
    predicate = getInitialPredicate(expression)
    newExpression = predicate + "(" + ",".join(
                    attributes[1:]) + ")"
    return newExpression
```

(2A)

AI - LAB - TEST - 2

REVANTH R
1BM18CS082

```
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 []
if isConstant(exp1):
    return [(exp1, exp2)]
if isConstant(exp2):
    return [(exp2, exp1)]
if isvariable(exp1):
    return [(exp2, exp1)] if not
    check occurs(exp2, exp1) else []
```

REVANTH.R
1BM18CS082

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 []

attribute count1 = len(getAttributes(exp1))

attribute count2 = len(getAttributes(exp2))

if attributeCount1 = attributeCount2:

print(f"Length of attributes {attribute

count1} and {attributeCount2} do not match.

Cannot be unified")

return []

(3A)

AI - TEST-2

REVANTH K
1BM18CS082

head 1 = Get First Part (exp 1)

head 2 = Get First Part (exp 2)

initial Substitution = unify (head 1, head 2)

if not initial Substitution: return {}

if attribute count 1 == 1:

return initial Substitution

tail 1 = get Remaining Part (exp 1)

tail 2 = Get Remaining Part (exp 2)

if initial Substitution != {}:

tail 1 = apply (tail 1, initial substitution)

tail 2 = apply (tail 2, initial substitution)

remaining Substitution = unify (tail 1, tail 2)

if not remaining substitution:

return {}

return initial substitution + remaining substitution.

(3B)

AI - LAB - TEST - 2

REVANTH. R

1BM18CS082

if __name__ == "__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 substitutions])

(AB)