Date __/__/__
Page _____
SPLASH

AI Lab-2

Pranjal Sinha
1BM18CS073

Prog-1

```
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

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

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

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

def getRemainingPart(expression):
    predicate = getInitialPredicate(expression)
    attributes = getAttributes(expression)
    newExpression = predicate + "(" + ",".join(attributes[1:]) + ")"

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")
```

Date __/__/__
Page _____
SPLASH

```
        return[]

    if isConstant(expl):
        return [(expl, exp2)]

    if isConstant(exp2):
        return [(exp2, expl)]

    if isVariable(expl):
        return [(exp2, expl)] if not checkOccurs
            (expl, exp2) else []

    if isVariable(exp2):
        return [(expl, exp2)] if not checkOccurs
            (exp2, expl) else []

    if getInitialPredicate(expl) != getInitialPredicate(exp2)
        print("Cannot be unified as they do not
            match")
        return []

attributeCount1 = len(getAttributes(expl))
attributeCount2 = len(getAttributes(exp2))
if attributeCount1 != attributeCount2:
    print(f"Lengths of attributes {attributeCount1}
        and {attributeCount2} do not match. Cannot be
        unified")
    return[]

head1 = getFirstPart(expl)
head2 = getFirstPart(exp2)
```

```
initialSubstitution = unify(head1, head2)
if not initialSubstitution:
    return []
if attributeCount1 == 1:
    return initialSubstitutions

tail1 = getRemainingPart(exp1)
tail2 = getRemainingPart(exp2)

if initialSubstitution != []:
    tail1 = apply(tail1, initialSubstitution)
    tail2 = apply(tail2, initialSubstitution)

remainingSubstitution = unify(tail1, tail2)

return initialSubstitution + remainingSubstitution

if __name__ == "__main__":

    print("Enter 1st expression")
    e1 = input()
    print("Enter 2nd expression")
    e2 = input()
    substitutions = unify(e1, e2)
    print(["/".join(substitution) for
        substitution in substitutions])
```