Program 8: Implement unification in first order logic

Write_up:

```
PRANAV JAGADEESH IBMI8CSO71 PROGRAM - 8
Implement unification in first order logic
     get Attributer (expression):
del get Initial Predicate (enforcirion):
      networ expression. Aftet ("(")[0]
def in Constant (char):
      return char ireffer () and len (char) = = 1
return char islower () and len (char) == 1

def replace. Attributer (ent), old, new):

attributer = get Attributes (ent)

for index, val in enumerate (attributes):

if wal == old:

attributes [index] = new
```



predicate = getInitialPredicate (exp)
neturn predicate + "(" + ", ". join (attributer) + ")"
def apply (exp, substitutions):
for substitution in substitutions
new, old = rulestibilion
 ent = replace Attributes (ent, old, new)
return exp
def sheek duurs (nar, exp):
if end (var) = = -1:
netwern false
return true
def get First Part (expression):
attributes = get Attributes (expression)
return attribute [D]
del getRemaining Part (expression):
briderale = get Intral Predicale (Expression)
attributes = get Attributes (enpression)
new Enfression = predicate + "("+","-join (altributes [I:]) +")"
Gilyun new crip secret
del unify (enf2, enf2):
y expl - gipt
if is Constant (exp1) and in Constant (exp2):
of a contant (mp) some manufactor (enpe)
if expl1 = exp2: geturn False
geturn laws.
return [(exp1; enp2)]
if in Constant [(enf2):
return [(exp2) exp1)]
if is Nariable (exp I):
il che bleever (exp2):
if checkOccurs (enp1, enp2): acturn False
June Com

	classmate
	Date Page
	Toda (1)
*	
	else:
	neturn [(exp2, exp1)]
	if isVariable (enf. 2).
	if check Occurs (exp2, exp 1):
	siction False
	else
	neturn [(exp2, expl)]
	if is Variable (exp2):
	if cheek Occurs (call ent):
	if sheek Occurs (cnf2, cnp 1): return False
	clse
	return ((ex/22, ex/2))
	if is Variable (exp2):
	if check Owers (enp2):
	neturn Falle
	de.
	neturn ((exp1, exp2))
	but all fied este (ent) != get duit ial Preduate (ex)?
	of get Initial Gredicate (ent) != get Initial Predicate (ext): print ("Inedicates do not match. Cannot be unified") neturn False
	this to the account
	rattribute Count 1 = len (get Attributes (enfst))
	attribute (ount 2 = len get Attributes (exp2))
	attribute Count 2 = len (get Attributes (exp2)) ef attribute Count 1 (= attribute Count 2:
	head 1 = get First Part (exp1)
	The second of th
	meteral Julestitulian = unify (head 1 hand 2)
	if not initial Substitution:
	neturn False
	if attributeCount 1==1:
	return initial suletitution
	lail 1 = get Remaining Part (enf 1)



*	Lail 2 = getkemaining Part (exp2)
	hlm aining ulutitution = mify (tail, tail2)
	if not remaining substitution:
	notrein tale
	initial Substitution entend (exemaining Substitution)
	outrum initial Substitution
	again = 'y'
	while again == 'y';
	bount ()
	ent 1 = input ("Enter first Expression")
	exp 1 = input ("Enter first Expression:") exp 2 = input ("Enter second Enfression:")
	print ("Original Enpremisons: ")
	frint (enp1)
	print (exp2)
	substitutions= unify (expl, exp2)
	print ("Sulestitutions")
	1, ct (P. l. later time)
	again = input ("Do you want to unify again & (Y/N): ")
	· capitalize ()
	(hiest()
	WTPUT: - Enter first expression: Knows (John, F(x))
	Enter alroad explantion. Knows (y, F(4(y))
	Original Expherions:
	Knows (John P(x))
	Knows (y, FCG(y))
-	Culutitution: ((Zohn', y'), (G (y', x'))
	Do you want to unify again 2 (Y/N): Y
/	the furt explanion. Thunan (mulling
/	Enter second emperción : Muman (Julius)
/	Original Explication: Human (Marcus)
/	Human (July)
/	Predicates do not match. Connet be unified Sulutitutions; False Poryon want to unify again & (Y/N). W
1	10 your man to mapy again a comp
	and the state of t

```
Program:
import re
def getAttributes(expression):
  expression = expression.split("(")[1:]
  expression = "(".join(expression)
  expression = expression[:-1]
  expression = re.split("(?<!\(.),(?!.\))", expression)
  return expression
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)
  for index, val in enumerate(attributes):
     if val == old:
       attributes[index] = new
  predicate = getInitialPredicate(exp)
  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):
  attributes = getAttributes(expression)
  return attributes[0]
def getRemainingPart(expression):
  predicate = getInitialPredicate(expression)
  attributes = getAttributes(expression)
  newExpression = predicate + "(" + ",".join(attributes[1:]) + ")"
  return newExpression
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 is Variable(exp2):
```

```
if checkOccurs(exp2, exp1):
       return False
     else:
       return [(exp1, exp2)]
  if getInitialPredicate(exp1) != getInitialPredicate(exp2):
     print("Predicates do not match. Cannot be unified")
     return False
  attributeCount1 = len(getAttributes(exp1))
  attributeCount2 = 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)
     tail2 = apply(tail2, initialSubstitution)
  remainingSubstitution = unify(tail1, tail2)
  if not remainingSubstitution:
     return False
  initialSubstitution.extend(remainingSubstitution)
  return initialSubstitution
again = 'Y'
while again == 'Y':
  print()
```

```
exp1 = input("Enter first Expression: ")
 exp2 = input("Enter second Expression: ")
 print("Original Expressions: ")
 print(exp1)
 print(exp2)
 substitutions = unify(exp1, exp2)
 print("Substitutions:")
 print(substitutions)
 again = input("Do you want to unify again? (Y/N): ").capitalize()
 print()
Output:
 Enter first Expression: Knows(John, F(x))
 Enter second Expression: Knows(y, F(G(y)))
 Original Expressions:
 Knows(John,F(x))
 Knows(y,F(G(y))
 Substitutions:
 [('John', 'y'), ('G(y', 'x')]
 Do you want to unify again? (Y/N): Y
 Enter first Expression: Human(Marcus)
 Enter second Expression: Human(Julius)
 Original Expressions:
 Human(Marcus)
 Human(Julius)
 Predicates do not match. Cannot be unified
 Substitutions:
 False
 Do you want to unify again? (Y/N): N
 Process finished with exit code 0
```