Output:



Kanjika Singh-1BM21CS086 $[\mbox{``animal(y)|loves(x,y)]\&[\mbox{``loves(x,y)|animal(y)]}}$ $[animal(G(x))\&\sim loves(x,G(x))]|[loves(F(x),x)]$ [~american(x)|~weapon(y)|~sells(x,y,z)|~hostile(z)]|criminal(x)

Program 10: Forward Reasoning

```
Code:
```

```
import re
def isVariable(x):
    return len(x) == 1 and x.islower() and x.isalpha()
def getAttributes(string):
    expr = '\([^)]+\)'
    matches = re.findall(expr, string)
    return matches
def getPredicates(string):
    expr = '([a-z^-]+)([^&|]+)'
    return re.findall(expr, string)
    class Fact:
    def __init__(self, expression):
        self.expression = expression
        predicate, params = self.splitExpression(expression)
        self.predicate = predicate
        self.params = params
        self.result = any(self.getConstants())
    def splitExpression(self, expression):
        predicate = getPredicates(expression)[0]
        params = getAttributes(expression)[0].strip('()').split(',')
        return [predicate, params]
    def getResult(self):
        return self.result
    def getConstants(self):
        return [None if isVariable(c) else c for c in self.params]
    def getVariables(self):
        return [v if isVariable(v) else None for v in self.params]
    def substitute(self, constants):
        c = constants.copy()
        f = f''(self.predicate)((','.join((constants.pop(0) if isVariable(p) else p for p in ))
self.params])})"
        return Fact(f)
class Implication:
```

```
def __init__(self, expression):
        self.expression = expression
        1 = expression.split('=>')
        self.lhs = [Fact(f) for f in 1[0].split('&')]
        self.rhs = Fact(l[1])
    def evaluate(self, facts):
        constants = {}
        new lhs = []
        for fact in facts:
            for val in self.lhs:
                if val.predicate == fact.predicate:
                    for i, v in enumerate(val.getVariables()):
                            constants[v] = fact.getConstants()[i]
                    new_lhs.append(fact)
        predicate, attributes = getPredicates(self.rhs.expression)[0],
str(getAttributes(self.rhs.expression)[0])
       for key in constants:
            if constants[key]:
                attributes = attributes.replace(key, constants[key])
        expr = f'{predicate}{attributes}'
        return Fact(expr) if len(new_lhs) and all([f.getResult() for f in new_lhs]) else None
class KB:
    def __init__(self):
        self.facts = set()
        self.implications = set()
    def tell(self, e):
        if '=>' in e:
            self.implications.add(Implication(e))
        else:
            self.facts.add(Fact(e))
       for i in self.implications:
            res = i.evaluate(self.facts)
            if res:
                self.facts.add(res)
    def query(self, e):
       facts = set([f.expression for f in self.facts])
        i = 1
        print(f'Querying {e}:')
        for f in facts:
            if Fact(f).predicate == Fact(e).predicate:
                print(f'\t{i}. {f}')
                i += 1
```

```
def display(self):
        print("All facts: ")
        for i, f in enumerate(set([f.expression for f in self.facts])):
            print(f'\t{i+1}. \{f\}')
print("Kanjika Singh-1BM21CS086")
kb = KB()
kb.tell('missile(x)=>weapon(x)')
kb.tell('missile(M1)')
kb.tell('enemy(x,America)=>hostile(x)')
kb.tell('american(West)')
kb.tell('enemy(Nono,America)')
kb.tell('owns(Nono,M1)')
kb.tell('missile(x)&owns(Nono,x)=>sells(West,x,Nono)')
kb.tell('american(x)&weapon(y)&sells(x,y,z)&hostile(z)=>criminal(x)')
kb.query('criminal(x)')
kb.display()
print("Kanjika Singh-1BM21CS086")
kb = KB()
kb_.tell('king(x)\&greedy(x)=>evil(x)')
kb_.tell('king(John)')
kb_.tell('greedy(John)')
kb_.tell('king(Richard)')
kb_.query('evil(x)')
```

Observation:

Date: 19 1/24
Forward Reasoning
The state of the s
exps = '([a-2] +)([^*1]) +()' return re fuidall (exps, string)
return re fuidals (exps, string)
class Implication:
def wit_ (sey, expression)
self the [fact (k) has f in 1 To). Shelt (b)7
sey-shs = fact((Ti))
Charles Charles and Shares
def evaluate (seef, facts):
Constants - 5 3
naw-lhy=()
for fact in facts:
for val in self. Us:
if val predicale = fact predicate
for i, v in enumerate (val. get varia (s).
My v:
constant [v) = feact get Constant.
new etrs append (fact)
Class Kb:
def ter (seef, e):
if (=) u'e
Self implication add (Implication(e))

Scanned with OKEN Scanner

print ("All feach:") for i, f in enumerate (set (f. enpression for i in cert facts))); print (f' (t ft +12, ffr)) kb +cul (king (x) & gredy(x) =) evrl(x)') kb +cul ('king (John)') kb +cul ('gredy (John)') kb +cul ('lking (Richard)') kb - query (kin (x)')	print ("All feach:") for 1, f in enumerate (set (f. enpression)) for f in cert facts)))); print (f' (t St +12, Sf2). Kb +cul (King (x) & gredy(x) =) evr'L (x)') Kb +cul ('King (John)') Kb +cul ('Gready (John)') Kb +cul ('Gready (John)')		def display (self):
Print (f'(t (t +12, (f))) (b = tB() (b +cul (King (x) & gredy(x) =) evil(x)) (b +cul (King (John)) (b +cul (King (John))) (b +cul (King (John)))	print (f'(t (t +12, f +4)) (b = tB() (b +cul (King (x) & gredy(x) =) eur'L(x)') (b +cul ('King (John)') (cb +cul ('gready (John)') (b +cul ('gready (John)')		brust ("All beion:")
Print (f'(t (t +12, (f))) (b = tB() (b +cul (King (x) & gredy(x) =) evil(x)) (b +cul (King (John)) (b +cul (King (John))) (b +cul (King (John)))	print (f'(t (t +12, f +4)) (b = tB() (b +cul (King (x) & gredy(x) =) eur'L(x)') (b +cul ('King (John)') (cb +cul ('gready (John)') (b +cul ('gready (John)')		Lox 1.1 in enumerate (set (f. enfects
Print (f'(t (t +12, (f))) (b = tB() (b +cul (King (x) & gredy(x) =) evil(x)) (b +cul (King (John)) (b +cul (King (John))) (b +cul (King (John)))	print (f'(t (t +12, f +4)) (b = tB() (b +cul (King (x) & gredy(x) =) eur'L(x)') (b +cul ('King (John)') (cb +cul ('gready (John)') (b +cul ('gready (John)')		Las I in cell facts) Ils
Kb + cel (King (x) & gredy(x) =) evr'(x)') Kb + cel (King (John)') Kb + cel (gready (John)') Kb + cel (gready (John)')	Kb = KB() Kb + cll (King (x) & gredy(x) =) evr'((x)') Kb + cel ('King (John)') Kb + cel ('gready (John)') Kb + cel ('gready (John)')	The state of	ne und (+ 1 t 5t + 1 2 - 5 t 3)
KAS tell ('gready (John)')	KAS tell ('gready (John)')		pure of the state
KAS tell ('gready (John)')	KAS tell ('gready (John)')		$r_{i} = r_{B(i)}$
KAS tell ('gready (John)')	KAS tell ('gready (John)')	LTW.	Kb. tell (King (x) & gredy(x) =) evil(x)
(As tell ('gready (sound)')	(As tell (greaty (sound))		
The Lett 1 King Kicher	The Left 1 King Kicker		KA Lead I Deaving Comment
Kb-query (kiri (x)')	Kb-query (Evil (2)")		The tell I King I Kichen
The state of	The standard of the standard o		Kb-query (Eir (a)')
That the first	the teat water		ad and sealed
	for Leas we lead to		That the Figure
		BE	Late was no seal that