Name - Sushrut Singh
USN - 1BM18CS094

1/1/202

AI LAB TEST-II

5.
```
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-zA-Z]+)\([^&|]+\)'
return re.findall(expr, string)

class fact:
    def __init__(self, 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() else for in
self.params]

def getVariables(self):
return [v if isVariable(v) else None for v in self.
params]

def getSubstitute(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
l = expression.split("=>")
self.lhs = [fact(f) for f in l[0].split("&")]

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()):
if v:
constants[v] = fact.get(constant())[i]
new_lhs.append(fact)
```

fact

predicate, attributes = set(predicates(self.rhs.expression)[0]), set(getAttributes(self.rhs.expression)[0])
for key in bindings:
attributes = attributes.replace(key, bindings[key])
expr = f'{predicate}{attributes}'
return fact(expr) if len(new - lhs) and all([fact.getNet() for fact 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 ask(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}')

fact

1 + = 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}')

def main()
    kb = KB()
print("Enter no of FOL expressions present in KB:")
    n = int(input())
    print("Enter the expressions:")
        for i in range(n):
        fact = input()
        kb.tell(fact)
        print("Enter the query:")
            query = input()
            kb.ask(query)
            kb.display()

main()
```

Sasad