AI LAB TEST-2

SHASHIDHAR PATIL
1BM18CS099

AI LAB-TEST-2

28/12/2020.

Program:

3. P.T 'SNOOPY will die" using resolution:

```
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 getPredicate(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
```

SHADUNHAR AAIB
10M1924049

```
self result = any (self get constants ())

def split expression (self expression):
    predicate = get predicate (expression)[0].
    params = get attributes (expression)[0]. strip('()') .split (',').
    return (predicate, params)

def get result (self):
    return self result

def get constants (self):
    return [None if is variable (c) else c for c in self.params]

def get variables (self):
    return [v if is variable (v) else none for v in
self .params]

def substitute (self, constants)
    c = constants .copy()
    f = f"{self .predicate }({','.join (constants.pop(0)
if is variable (p) else p for p in self. params])})"
    return Fact(f)
```

Rule Implication:

```
def __init__(self, expression):
    self.expression = expression
    ln = expression.split("=>")
    self.lhs = [Pattern(i) for i in ln[0].split("&")]
    self.rhs = Pattern(ln[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()):
                    if v: constants[v] = fact.getConstants()[i]
                new_lhs.append(fact)
    predicate, attributes = getPredicate(self.rhs.expression)
    return str(getAttributes(self.rhs.expression)[0])
```

03

```
for key in constant:
    if constant(key):
        attribute = attribute.replace(key, consts(key))
        expr = f'{predicate} {attribute}'
        ~~expr = f'{pred...~~
        return fact(expr) if len(new_lhs) and all
        (f.get_result() for f in new_lhs) else None

class KB:
    def __init__(self):
        self.facts = set()
        self.rules = set()

    def query(self, e):
        facts = set(f.constraint 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}')

    def display(self):
        print("All facts:")
        for i, f in enumerate(set([f.constraint for f in self.facts])):
            print(f'\t{i+1}. {f}')
```