AI- Writeup

Khaithik.S.Anand
1BM18CS046
AI-Lab - Test -2
29/12/2020

Program:

```
import re
def isVariable(x):
    return len(x) == 1 and x.islower()

def getAttributes(string):
    return re.findall('\([^)]+\)', string)

def getPredicates(string):
    return re.findall('([a-z~]+)\([^&]+\)', string)

class Fact:
    def __init__(self, expression):
        self.expression = expression
        predicate, params = self.splitExpress(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
```

AI-Writeup

Khaithik. S. Anand
1BM18CS046
AI-Lab-Test-2
29/12/2020

```
    def getConstant(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]

class Implication:
    def __init__(self, expression):
        self.expression = expression
        l = expression.split('=>')
        self.lhs = [Fact(f) for f in l[0].split('&')]
        self.rhs = Fact(l[1])

    def evaluate(self, facts):
        constants = {}
        new_lhs = []
        for fact in facts:
            for val in self.lhs + [self.rhs]:
                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 = self.rhs.predicate, '('+','.join(self.params)+')'
```

AI- Write up

Khushik . S. Anand
1BM18CS046
AI - Lab - Test - 2
29/12/2020

```
        for key in constants:
            if constants[key]:
                attributes = attribute.replace(key, constants[key])
        exps = f'{predicate}({attribute})'
        return Fact(exps) if ten(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(Implications(e))
        else:
            self.facts.add(Fact(e))
        for i in self.implications:
            res = i.evaluate(self.facts)
            if res:
                self.facts.add(res)
```


Khushik

AI - Write up

Khaithik - S. Anand
IBM18CS046
AI - Lab test - 2
29/12/2020

```
def query(self, e):
    for i in self.implications:
        res = i.evalute(self.facts)
        if res:
            self.facts.add(res)
    facts = set([f.expression for f in self.facts])
    i = 1
    print(f'Querying {e}:')
    for f in facts:
        if Facts(f).expression == Fact(e).expression:
            print(f'The query {e} is satisfied')
            return
    print(f'The query {e} is refuted')

def display(self):
    for i in self.implications:
        res = i.evaluate(self.facts)
        if res:
            self.facts.add(res)
    print("All facts:")
    for i, f in enumerate(set([f.expression for f in self.facts])):
        print(f'\t{i+1}. {f}')
```

AI - Write up

Khithik . S . Rao
IBM18CS046
AI - Lab - Test-2
29/12/2020

```
kb = KB()
kb.tell('food(x) => Likes(x, Rani)')
kb.tell('food(Peanut)')
kb.tell('~food(Mug)')

kb.query('likes(Peanut, Rani)')
print()
kb.display()
```

Khithik