Date

Expt. No. Page No.

Nidhish Lakkimana
1BM18CS079

```
import re
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]

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:
                if val.predicate == fact.predicate:
```

Teacher's Signature :

```
                for i, v in enumerate(val.getvariables()):
                    if v:
                        constants[v] = kafc.getconstants()[i]
                new_lhs.append(facts)

    predicate, attributes = getpredicates(self.rhs.expression()[0], 
        string(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 facts(f).predicate == fact(e).predicate
            print(f'\t{i}. {f}')
            i += 1.
```