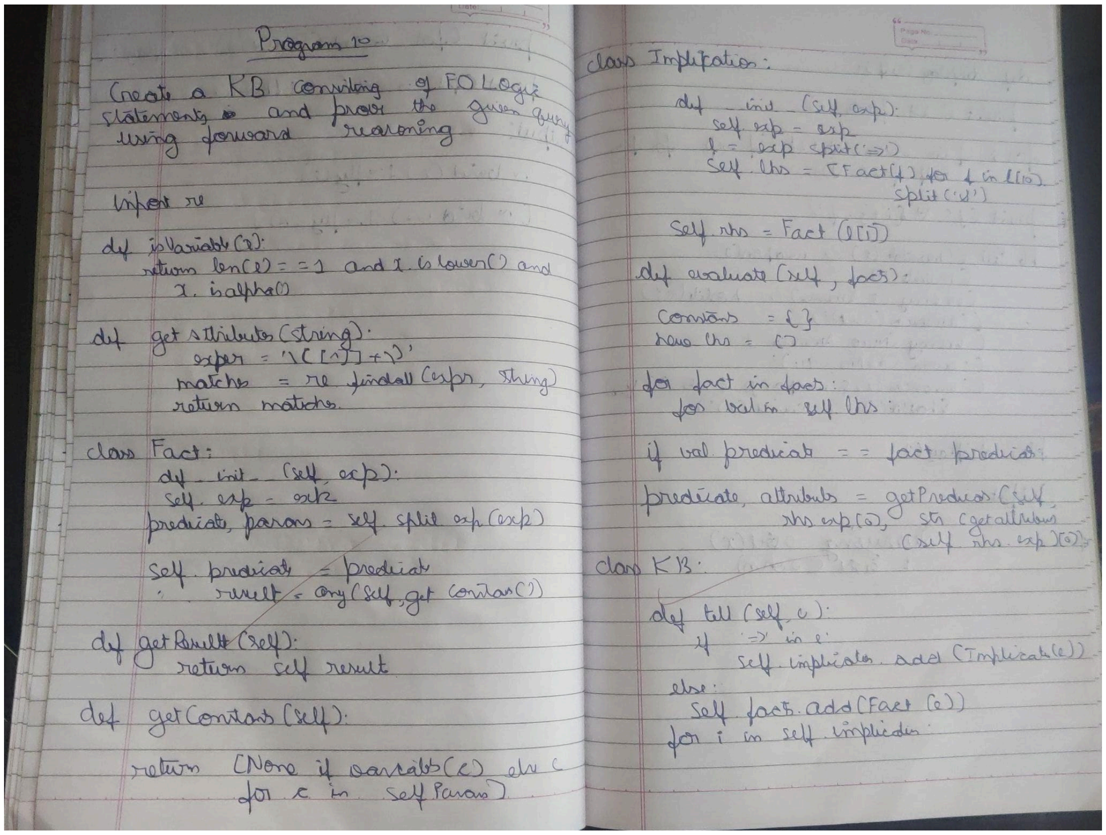
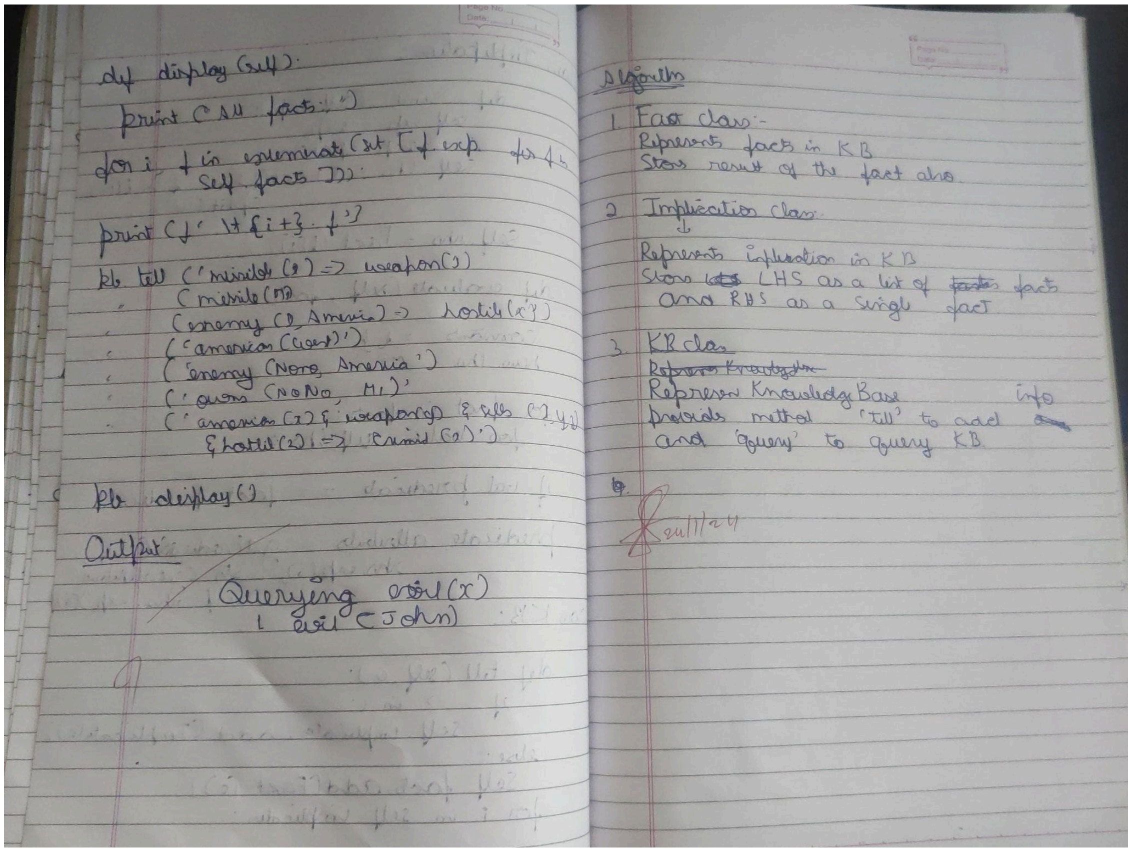


Program 10

Create a KB consisting of FO Logic statements and prove the given query using forward reasoning

```
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

class Fact:
        def __init__(self, exp):
        self.exp = exp
        predicate, params = self.splitExp(exp)

        self.predicate = predicate
            result = any(self.getConstants())

def getResult(self):
        return self.result

def getConstants(self):

    return [None if isVariable(c) else c
            for c in self.params]
```

```
class Implication:

        def __init__(self, exp):
            self.exp = exp
            l = exp.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

    predicate, attributes = getPredicates(self.
                rhs.exp[0], str(getattributes
                            (self.rhs.exp)[0])

class KB:

    def tell(self, e):
        if '=>' in e:
            self.implications.add(Implication(e))
    else:
        Self.facts.add(Fact(e))
    for i in self.implications
```



```
def display(self):
    print("All facts: ")

for i, f in enumerate(set([f.exp for f in
        self.facts])):

print(f'\t{i+1}. {f}')

kb.tell('missile(x) => weapon(x)')
  "     ('missile(M1)')
  "     ('enemy(x, America) => hostile(x)')
  "     ('american(West)')
  "     ('enemy(Nono, America)')
  "     ('owns(NoNo, M1)')
  "     ('american(x) & weapon(y) & sells(x,y,z)
            & hostile(z) => criminal(x)')

kb.display()
```

Output

```
Querying evil(x)
 1 evil(John)
```

Algorithm

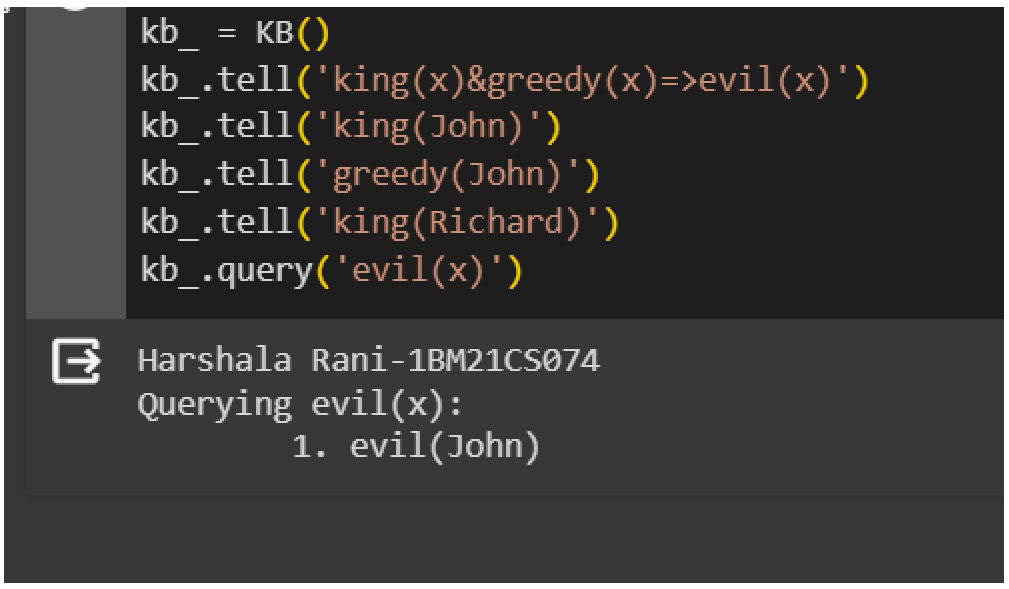
1. Fact class:-
   Represents facts in KB
   Stores result of the fact also.

2. Implication class:
   Represents implication in KB
   Stores ~~[illegible]~~ LHS as a list of ~~[illegible]~~ facts and RHS as a single fact

3. KB class
   ~~Represent Knowledge~~
   Represent Knowledge Base info
   provides method 'tell' to add ~~[illegible]~~ and 'query' to query KB.

4.

[illegible] 24/11/24



```
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)')
```

```
Harshala Rani-1BM21CS074
Querying evil(x):
        1. evil(John)
```