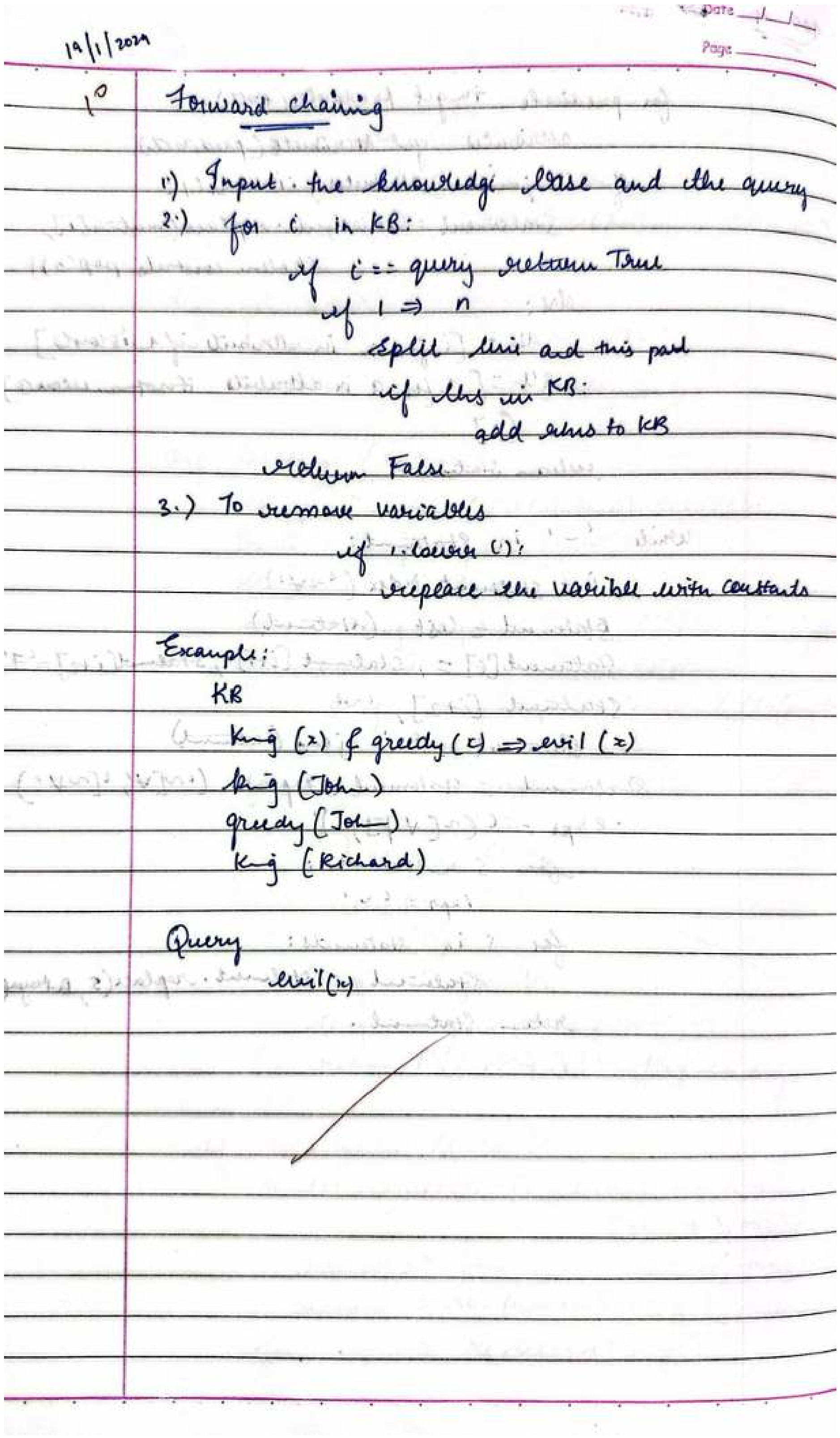
Forward chaining
Code:-



19/1/2024

Date / /
Page

1° Forward chaining

```
1.) Input the knowledge base and the query
2.) for i in KB:
        if i == query return True
            if l ⇒ n
                split lhs and rhs part
                if lhs in KB:
                        add rhs to KB
    return False
3.) To remove variables
            if i.lower ():
                replace the varible with constants
```

Example:

KB
- King (x) & greedy (x) ⇒ evil (x)
- King (John)
- greedy (John)
- King (Richard)

Query
evil(x)

Code :-

```
import re
def is variable(x):
    return len(x)==1 and x islower() and x.isalpha()

def getAttributes(String):
    exper = '\([^]]+\)'
    matches = re.findall(expr. String)
    return matches.

def get predicates (String):
    expr = '([a-z~]+)\([^&[]+\)'
    return re. findall (expr. String)

class Fact:
    def __init__(self - expression).
        self. expression = expression.
        self. params = params
        self . result = any(self . getconstants())

    def split Expression(self , expression):
        predicate = get predicates(expression)(0)
        params = get Atributes (expression[0])
                   Strip ('()'). split (',')
        return [predicate , params)

    def getresult(self):
        return self.result.

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

```
def get variable(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
                  if val_predicate = fact_predicate
                     for i,v in enumerate(val.getvariables())

Class KB:
    ~~def __init__(self):~~
        ~~self.facts = set()~~
        ~~self.implications = set()~~

    def tell (self.c)
        if => in c:
            self.implications.add(Implication

        for i in self.implications:
            res = i.evaluate (self.facts)
```

Date __/__/__

Page ______

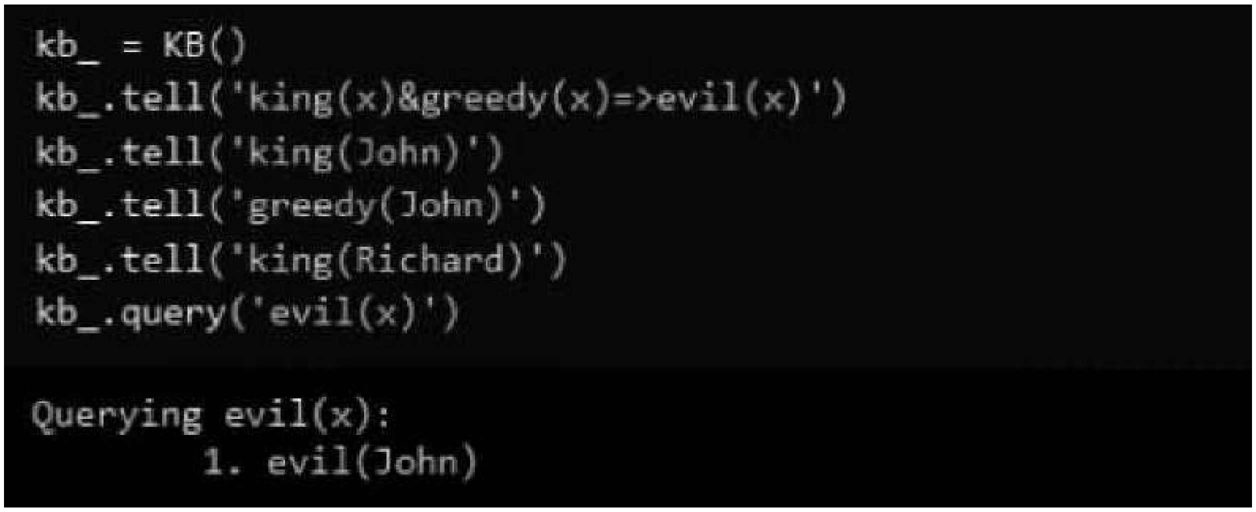
```
def query (self . q):
    facts = set ([f.expression for in self.facts]

def display (self):
    print (" All facts")
        for i, in enumerate (set ([f.expression)
                for in self. Facts
            print (f"{i+1} {f}")

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)')
```

Output:

```
Querying evil(x):
        evil (John)
```

[illegible] 24/1/24
Complete!

Output:-



```
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)')
```

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