I sem.

C.S.
AI LAB TEST-2

SOWMYA.D.U
1BM19CS408

Question - 5th

Forward reasoning system to prove query "someone who are intelligent cannot read" using forward chaining.

Program

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

class Fact :
  def_init_(self, expression):
     self.expression = expression
     predicate, params = self.splitExpression (expression)
      self. splitExpression (expression)
      self. predicate = predicate
      self. params = params
      self. result = any (self. getConstants())
```

[illegible]

Sowmya A. W. J.
1BM19CS407

```
def splitExpression (self, expression) :
predicate = getPredicates (expression) [0]
 params = getAttributes (expression) [0].strip ('()').split(',')
  return[predicate , params]

def ~~splitExpression (self, expression):~~
 ~~predicate = getPredicates (expression)[0]~~
   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)
```

2

[illegible]

SOWMYA.D.
1BM19CS40

```
ss Implication:
 ef _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()):
                        if v:
                    constants[v] = fact.getconstants()[i]
                    new-lhs.append(fact)
                    predicate, attributes =
                    get Predicates(self.rhs.expression)[0],
                    str(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.
```

[illegible]

SOWMYA.D
1BM19CS4

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

4

[illegible]

SOWMYA .Ŵ.U
IBM19CS408

```
def main ( ):
   kb = KB( )
  print ("Enter KB : (enter e to exit )")
   while True :
     t = input ( )
      if (t== 'e'):
           break
      kb.tell(t)
    print ("Enter Query:")
    q = input()
    kb query (q)
     Kb. display ( )
```

5

[illegible]