# AI LAB

VAISHNAV.S
1BM18CS121
C Section
28/Dec/2020

1 Snoopy will die
a "Snoopy is a dog
b "all dogs are animals
c "all animals will die

```
import re

def isVariable(x):
        return len(x) == 1 and x.islower.
c)   and x.isalpha()

def getAttributes(String:)
      Expr = '\([^]]+\)'
      match = r.findall(expr, string)
       return matches.

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

In[ ]
```

```
Class fact :
        def _initit_ ( Self , expression ):
            self . expression= expression predicat ,
Params = self . spl .
 it expression (expression )
            self . predict = predicat
            self . params = Params .
            self . result = any ( self . get cons ( ) )
      def split expression ( Self , expres sion ):
      Predecat = get Predecats (ex pression [0] .
       Params = get Attributs (expression ) [0] .
    Strip ('()') split (',')
            return [ Predecit , Params ]
          def get result ( self ):
              return self . result .
        def get constant ( self ):
            return [ None if is Variabl
  (c) else c for c in self Parms ] .
```

```
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 evaluat (Self, Facts):
        constants = {}
        new_ lhs = []
        for fact in facts :
            for val in Self . lhs :
                if val . Predicat ==
    Fact . Predict :
                    for i , v in enumerat (val. get,
                                        Variable ());
                                                   if v:
    S[v] = fact . get Constant () [i]
                                new _lhs . append(fact).
        for key in constants :
            if constant [key]:
```

```
attributes = attributes 6 . replace (key )

    return Fact (exps) if len(new _lhs ) and all (f.
  get Result () for f i n new - lhs ] else None.


    class KB

        def _init_( self,e):
            if '=>' in e :
                self. implication 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
  v is f in self . facts ] ]
            i = 1
            print (f'querya {e}:1)
              for(f '. Querying { e }: ').
        if fact (f.) predict == Fact (e). predict .
```

```
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}')
```