Ser BOUND AILABTEST-2 IBMI9CS	6.1
Question 5th	
Forward reasoning system to prove a parry "sameone	
who are intelligent cours t read " using forward	
chaining.	
- Korts or is C. D. C. washing and December of the	
Program 10 100 percent komment in the levour	
impour ve	
def is Variable (x):  veturn len (x) = = 1 and x. islower() and	
x. isalpha ()	
de ger Attributer Cstring 1/2 / 200 mosts 1 hr 600 mg	
1 expor = 0 (65) 1 = 1 (100 11 (1 = 600 11 = 20) & = 0 000 11 =	
matches = re, produi (expr., swing)	
oreturn matches	
deg get Rnedicates Cstring): exper = 'C[a-2~J+)/([^4])+1)'	
retur re j'raul (expr, string)	
claus Face:	
sey. expression = expression	
predicate, params = sey. speit Expression (expression	n)
sey. Split Expression (expression)	
self. predicate = predicate	
Self. parame = parame.	
sey, oresult = any (sey, get (onstants())	
J. Commy.	5-0

Scanned with CamScanner

Scanned with CamScanner

params = get Priedicates (expression) [0], Staip ('()') spector params = get Attributes (expression) [0], Staip ('()') spector params de septembron (self), expression):  priedicate = get Priedicates (expression) [d]  params  de get Result (self):  return self result	def split expression (sey, expression):
params = get Attribute (expression):  Usturn (predicate, params)  deg Spritexpression (sely, expression):  predicate = get Predicates (expression) [d]  params  deg get Resout (sely):  voturn self resout  deg get Constanta (sely):  voturn [None of it Nariowse (c) else e for c in segl. params  deg get voliables (sely):  voturn [v ij is Nariowse (c) else e for v in  self: params]  deg substitute (self):  (= constants.copy()  y = y i f suf. predicate f  (f', '. join ([constants.pop(o); is variowse (p) else  voturn Fact (y)	
de spritexpression (sely, expression):  predicate: get Predicates (expression) TdJ  params  del get Resout (self):  voturn self resout  del get Constanta (self):  voturn [none: fix variable (e) else e for c in self params  del get variables (self):  voturn [v ] is variable(v) else none for vin  self. params]  del substitute (self):  (= constants .copy()  y= y'd self. predicate f  (f','.join ([constants.pop(o): fix variable (p) else  p for pin self. params]);)"  vuturn Fact (y)	params = get Attributer (expression) (0), Strip (0)
del get Constante (self):  vietur self various (c) eur e for c in self par  del get various et self):  vietur [v ij is various c(v) eur noné for vin  self. params]  del substitute (self);  (= constants.copy ()  y= y'd self.predicate)  (f','.join ([constants.pop(o): jis various e (p) eur  p for pin self.posams])));  vietur Fact (y)	retuent predicate, paranas
del get Constante (self):  vietur self various (c) eur e for c in self par  del get various et self):  vietur [v ij is various c(v) eur noné for vin  self. params]  del substitute (self);  (= constants.copy ()  y= y'd self.predicate)  (f','.join ([constants.pop(o): jis various e (p) eur  p for pin self.posams])));  vietur Fact (y)	de Splitexpression (self, expression):
del get Constante (self):  vietur self various (c) eur e for c in self par  del get various et self):  vietur [v ij is various c(v) eur noné for vin  self. params]  del substitute (self);  (= constants.copy ()  y= y'd self.predicate)  (f','.join ([constants.pop(o): jis various e (p) eur  p for pin self.posams])));  vietur Fact (y)	parans
ad get Constants (self):  vietuen [None 1] 'x Naviable (c) eue e dor c in self. para  dy get variables (self):  vietuen [v ij is naviable(v) eue none dor vin  self. params]  def substitute (self), constants ):  (= constants.copy()  y= y" d self. predicate f  ([i','.join ([eonstants.pop(o); [is variable (p) eue  p dor pin self. params])])"  vietuen Fact (y)	de ger Result (seef):
de get variables (self):  vieturn [v ] is variable(v) euro none fort vin  sey: params]  de substitute (self, constants):  (= constants.copy()  y= y" d sey. predicate f  (f','.join ([constants.pop(o); fis variable (p) euro  p for pin self. params]);)"  veturn Fact (y)	outur selforesult
de get variables (self):  vetuer [v ] is variable(v) eve none for vin  sey: params]  de substitute (self) constants):  (= constants.copy()  y= y" d sey.predicate f  (f','.join ([constants.pop(o); fix variable (p) eve  p for pin self.params]);)"  vetuer Fact (y)	de get constants (sey):
de get variables (self):  vetuer [v ] is variable(v) eve none for vin  sey: params]  de substitute (self) constants):  (= constants.copy()  y= y" d sey.predicate f  (f','.join ([constants.pop(o); fix variable (p) eve  p for pin self.params]);)"  vetuer Fact (y)	vieture Mone if is variable (c) elle 2 gor C 17 seff. para
Sey: params)  del substitute (sey; constants):  c = constants.copy()  y = y " d sey.predicate j  ( { ' , ' . join ([constants.pop(0):] is variable (p) else  p for pin sey.params])))"  veturn Fact (y)	de get variables (self):
Sey: params)  del substitute (sey; constants):  c = constants.copy()  y = y " d sey.predicate j  ( { ' , ' . join ([constants.pop(0):] is variable (p) else  p for pin sey.params])))"  veturn Fact (y)	outer [v ij is variable (v) ever vone jor v in
C= constant.copy()  q= y"d seq.predicate y  (q','.join ([eonstants.pop(o):] is variable (p) else  p doe pin seq.paname])))"  return Fact (q)	Sey: parame)
C= constant.copy()  q= y"d seq.predicate y  (q','.join ([eonstants.pop(o):] is variable (p) else  p doe pin seq.paname])))"  return Fact (q)	del culostitute (sell consent
( f', '.join ([eonstans.pop(o); is variable: (p) else p for pin seep.parama])))"  vetuen Fact (g)	c = constants.copy()
return Fact (g)	y= y"d sey. prédicate y
return Fact (g)	(f', '. join (Teonstant, projo) :
cretion Fact (g)	b dae bie son boarant])?)"
	return Fact (y)
$\mathcal{L}$	
THE STREET STREE	$\mathcal{L}$

1)210 . \_\_\_\_

```
Implication:
of _init _ (seef, expression):
 Self. expression = expression
  1 = expression.spair('=>')
  Seef. the = [Fact (g) for fin 100] - spect ('&')]
  sey. she = Fact (1[1])
   del evaluate (seef, facts):
     constants = {}
      hew-Ihs= []
     goe fact in facts:
        for val in seel. 1/2:
            ? val. predicate = = face. predicate:
                for is vin enumerate (val. get variables());
            constants()[;]
              new-Ihs. append (gas)
              paredicate, attributes =
              get Predicates (cely. orhos expression ) [0],
              Sta (get Attributes (seep. who. expension) [0])
            gas key in constants:
                if constants [ Key]:
     attibutes = attributes . replace (key, constants [key])
          exper = of depreciate by addibutes y'
          return Fact (expr)
           of len (new-lhs) and au ([g. get Result) yor yin
           new-IhSJ) ese
            Mone.
```

```
clar KB:
   del- wit - czord ):
     Seef. pach = set ()
     sey. implications = set ()
   de tell (seef, e)
     ? "=>"ine:
   Sey. inpercations. add (Implication (e))
       Sey. Jack. add (Fact (e))
      for i in selimpuications:
       vres = i. evouvate (sey, jacts)
       ژا محد
          sey gacts. and Cores
   det drand czeel se).
    gacta = set (Tg. expensión gor gin seg, facts )
      prine (q'queryingles:')
       for fin focts.
      if Fact ( g ). prédicate = = Fact (e). prédicate:
           ('d pp. dig =1' pring
   de dispeau ( sey):
       print ("AU facts:")
       par P, g'in enumerate C set C [g. exporention gos ginsey
                                 Jact I)).
           priva (811 + 91+ 16 4 9 priva
```

10M 10K2108

def main ():

kb = KB()

print ("Enter KB: (enter e to exit )")

while Torre:

t = input()

if (t == 'e'):

break

Kb. tell(t)

print ("Enter query:")

q = input()

kb query(q)

Kb. display()