Manas Agarwal
1BM18CS052
AI

Question 4

```
import re
def getAttributes (string):
    expr = '\([^)]+\)'
    matches = re.findAll(expr, string)
    return [m for m in str(matches) if m.isAlpha

def getPredicates (string):
    expr = '[a-z~]+\([A-Za-z,]+\)'
    return re.findAll(expr, string)

def deMorgan(sentence):
    string = ''.join(list(sentence).copy())
    string = string.replace('~~', '')
    flag = '[' in string
    string = string.replace('~[', '')
    string = string.strip(']')
    for predicate in getPredicates(string):
        string = string.replace(predicate,
                    f'~{predicate}')
```

```
s = list(string)

for i,c in enumerate(string):
        ~~string = string.replace(predicate,~~
        if c == 'V'
              s[i] = 'Λ'
        elif c == 'Λ'
              s[i] = 'V'

        ~~string = ''.join~~(s)

string = ''.join(s)
string = string.replace('~~', '')
return f'[{string}]' if flag else string

def skolemization( sentence):
     skolem_constants = [f'{chr(c)}' for c in
                       range(ord('A'), ord('Z')+1)]
     statement = ''.join(list(sentence).copy)

     matches = re.findAll('[∀∃].', statement)

     for match in matches[::-1]:
          statement = statement.replace(match, '')
          statements = re.findAll('\[\[[^]]
                              +\]]', statement)
```

```
for s in statements:
    statent = statement.replace( s, s[1: -1])
for  predicate in getPredicates (statement):
    attributes = getAttributes (predicate)
    if ''.join (attributes). islower():
        statement = statement.replace
        (match[1], skolem_constants.pop(0))

    else:
        al = [a for a  in attributes
                if a.islower()]

        aU = [a for a in attributes
                if not a.islower()]
                    [0]

        statement = statement .replace
        (aU, f'{sklem_constants.pop(0)}
    ({aL[0] if len[aL) else match[1]
        })')

return statement
```

```
def   fol to CNF (fol):
        statement = fol. replace ("<=>", "_")
        while '_' in statement:
            i = statement. index('_')
            new_statement = '[' + statement [: i]
              + '=>' + statement [i+1 :] +
             '] ∧ [' + statement [i+1 :] +
              '=>' + statement [: i] + ']'
        statement = statement. replace ("=>", '-')
        expr = '\[([^]]+)\]'
        statements = re. findAll (expr, statement)
        for i, s in enumerate (statements):
            if '[' in s and ']' not in s :
                statements [i] += ']'

        for s ~~in~~ in statements :
            statement = statement. replace (
                                  s, fol_to_CNF(S))
```

```
while '-) in statement:
                .index
    i = statements('-')
    b1 = statement.index('[') if '[' in
         statement else 0
    new_statement = '~' + statement[b1:i]
         + 'V' + statement[i+1:]
    statement = statement[:b1] +
         new_statement if b1>0 else
         new_statement.

while '~V' in statement:
    i = statement.index('~V')
    statement = list(statement)
    statement[i], statement[i+1],
        statement[i+2] = '∃', statements[i+2]
                  '~'
    statement = ''.join(statement)
```

```
while '~∃' in statement :
        i = statement.index('~∃')
        s = list(statement)
     s[i], s[i+1], s[i+2] = '∀', s[i+2], '~'
        statement = ''.join(s)
  statement = statement.replace('~[∀', '[~∀')
  statement = statement.replace('~[∃', '[~∃')
  expr = '(~[∀|∃].)'
  statements = re.findAll(expr, statement)
  for s in statements :
        statement = statement.replace(s, fol to cnf
                                                  (s))
  expr = '~\[[^]]+\]'
  statements = re.findAll(expr, statement)
  for s in statements :
        statement = statement.replace(s,
                          DeMorgan(s))
  return statement

fol = input("Enter FOL")
print ( Skolemization( fol to cnf(fol))
```