Nalesh Gangopadhyay
IBM18CS058

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):
        if c == 'V':
            s[i] = '^'
        elif c == '^':
            s[i] = 'V'
```


Nub

Neelesh Gangopadhyay
1BM18CS058

```
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
        statement = 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'{SKOLEM_CONSTANTS.pop(0)}({aL[0] if len(aL) else match[1]})')
    return statement
```

Neelesh Gangopadhyay
1BM18CS057

```
def fol_to_cnf(fol):
    Statement = fol.replace('<=>', "-")
    while '-' in Statement:
        i = statement.index('')
        new_statement = '[' + statement[:i] + '=>' + statement[i+1:]
        + ']^[' + statment[i+1:] + '=>' + statement[
          :i] + ']'
        Statement = new_statement
    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 statements
        Statement = statement.replace(s, fol_tocnf(s))
    while '-' in statement:
        i = statement.index('-')
        br = statement.index('[') if '[' in statement else 0
        new_statement = '~' + Statement[br:i] + 'V' + Statement[i+1]
        Statement = Statement[:br] + new_statement if br>0
            else new_statement
    while '~V' in statement:
        i = statement.index('~V')
        statement = list(statement)
```


Neelesh

Nalin Gangopadhyay
1BM18CS058

```
Statement[i], Statement[i+1], statement[i+2] = ']', statement[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, Demorgan(s))
    return Statement

fol = input("Enter F.O.L Statement \n")
  print("\nThe CNF Form is: ")
 print(Skolemization(fol_to_CNF(fol)))
```


Nalin