Pranav Arora
1BM18CS072

A.4 First Order Logic to Conjuctive Normal Form

```
import re

def getAttributes (String):
    expr = '\([^)]+\)'
    matches = re.findall (expr, string)
    return [m for m in str(matches) if m.isalpha()]

def getPredicates:
    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'{pred
    S = list (String)
    for i,c in enumerate(string):
        if (c == 'V'):
            S[i] = '^'
        elif ('c == '^')
            S[i] = 'V'
```

Pranav

```
    string = ''.join(s)
    string = string.replace("\n", "")
    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
                    (a, f'{SKOLEM.pop(0)}')
    return statement
```

Pass

```
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 statements:
        statement = statement.replace(s, fol_to_cnf(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 '~(' in statement:
    i = statement.index('~(')
    s = list(statement)
    s[i], s[i+1], s[i+2] = '~', s[i+2], '('
    statement = ''.join(s)

for s in statements:
    statement = statement.replace('~(', '(~')

for s in statements:
    statement = statement.replace(s,
                DeMorgan(s))

return statement

for = input()
print(Skolemization(Fol_to_CNF(fol)))
```