Date:
01/01/2021

AI Lab test-2

Revanth.NM
IBM18CS081

Program:

Convert the following English sentence into FOL. Then convert FOL statement into CNF and use input as

All gardeners like the Sun.

Algorithm:

```
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~]+\([A-z a-z ,]+\)'
    return re.findall (expr, string)
def DeMorgan (sentence)
    string = ''.join (list (sentence).copy())
        string = string.replace ('~~', '')
        string = string.strip ('[')
        string = string.strip (']')
    for predicate in getpredicate (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'
    string = ''.join(s)
    string = string.replace('~~', '')
    return f'~{string}' if flag else string

def skolemization(sentence):
    SKOLEM_CONSTANTS = [f'{chr(c)}' ]
    statement = ''.join(list(sentence).copy())
    matches = re.findall('[∀∃].', statement)
    for match in matches[::-1]:
        statement = statement.replace(match, '')
        statement = 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'{SOKLEM_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[p+1:] + 'J^[' +
                statement[p+1:] +'=>' + statement
                                        [:i] + 'J'

statement = new_statement
statement = statement.replace("=>","_")

expr = '\[([^]]+)\]'
statements = re.findall(expr, statement)
for i,s in enumerate(statements):
    if '[' in s and 'J' not in s:
        statement[i] += 'J'

for s in statements:
    statement = statement.replace(s, fol_to_cnf(s))
    while '_' in statement:
        i = statement.index('_')
        bi = statement.index('[') if '[' in statement else 0
        newstatement = '~' + statement[bi:i] + 'V' + statement[i+1:]
        statement = statement[:bi] + new_statement
                    if bi>0 else new_statement

while '~∀' in statement:
    i = statement.index('~∀')
    statement = list(statement)
    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('~[^', '[~^')
statements = re.findall(expr, statement)
for s in statements:
    statement = statement.replace(s, fol_to_cnf(s))
expr = '~\[[^]]+\]'
statements = re.findall(expr, statement)
for i in statements:
    statement = statement.replace(i, DeMorgan(i))
return statement.
fol_to_cnf(fol)
```