Hemanth Kumar V.C.
1BM18CS037
Sec 'A'

AI - LAB - TEST 9

Program 3:- Resolution.

Code:-

```
def _resolve (arg_one , arg_two):
    resolved = false
    s1 = make_sentence (arg_one)
    s2 = make_sentence (arg_two)
    resolve_s1 = None
    resolve_s2 = None
for i in s1:
    if isNotList (i):
        a1 = i[1].
        a1_not = true
    else :
        a1 = i
        a1_not = false
    for j in s2:
        if NotList (j):
            a2 = j[1]
            a2_not = true
        else a2[j]
            a2_not = false
        if a1 = = a2:
            if a1_not ! = a2_not :
            if resolved :
            return false
        else
            resolved = true
            resolve_s1 = i
            resolve_s2 = j
```

```
        break
    if not resolved:
        return False
    s1.remove(resolve_s1)
    s2.remove(resolve_s2)
    result = clear_duplicate(s1+s2)
    if len(result)==1:
        return result[0]
    elif len(result)>1:
        result.insert(0,'or')
    return result
def make_sentence(arg):
    if isLiteral(arg) or isNotList(arg):
        return [arg]
    if isOrList(arg):
        return clear_duplicate(arg[1:])
    return
def isClause(sentence):
    if isLiteral(sentence):
        return true
    if isNotList(sentence):
        if isLiteral(sentence[1]):
            return True
        else:
            return False
    if isOrList(sentence):
        for i in range(1, len(sentence)):
            if len(sentence[i])>2:
                return False
            elif not isClause(sentence[i]):
                return False
        return true
    return False
```

Hemanth Kumar V.C
1BM18CS037
Sec A

```
def isCNF(sentence):
    if isClause(sentence):
        return true
    elif isAndList(sentence):
        for s in sentence[1:]:
            if not isClause(s):
                return false
            return true
        return false
def negation(sentence):
    if isLiteral(sentence):
        return ['not', sentence]
    if isNotList(sentence):
        return sentence[1]

    if isAndList(sentence):
        result = ['or']
        for i in sentence[1:]:
            if isNotList(sentence):
                result.append(i[1])
            else:
                result.append(['not', sentence])
        return result
    if isOrList(sentence):
        result = ['and']
        for i in sentence[:]:
            if isNotList(sentence):
                result.append(i[1])
            else:
                result.append([['not',i]])
```

```
        return result
    return None
def convert_cnf(sentence):
    while not is_cnf(sentence):
        if sentence is None:
            return None
        sentence = make_cnf(sentence)
        return sentence
```