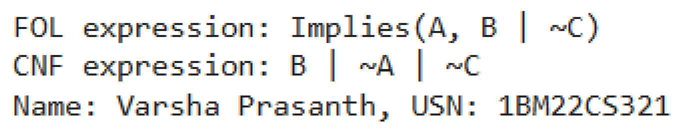
First Order Logic to Conjunctive Normal Form

```
from sympy.logic.boolalg import Or, And, Not, Implies, Equivalent
from sympy import symbols

def eliminate_implications(expr):
    """Eliminate implications and equivalences."""
    if isinstance(expr, Implies):
        return Or(Not(eliminate_implications(expr.args[0])),
                  eliminate_implications(expr.args[1]))
    elif isinstance(expr, Equivalent):
        left = eliminate_implications(expr.args[0])
        right = eliminate_implications(expr.args[1])
        return And(Or(Not(left), right), Or(Not(right), left))
    elif expr.is_Atom:
        return expr
    else:
        return expr.func(*[eliminate_implications(arg) for arg in
expr.args])

def push_negations(expr):
    """Push negations inward using De Morgan's laws."""
    if expr.is_Not:
        arg = expr.args[0]
        if isinstance(arg, And):
            return Or(*[push_negations(Not(sub_arg)) for sub_arg in
arg.args])
        elif isinstance(arg, Or):
            return And(*[push_negations(Not(sub_arg)) for sub_arg in
arg.args])
        elif isinstance(arg, Not):
            return push_negations(arg.args[0])
        else:
            return Not(push_negations(arg))
    elif expr.is_Atom:
        return expr
    else:
        return expr.func(*[push_negations(arg) for arg in expr.args])

def distribute_ands(expr):
    """Distribute AND over OR to obtain CNF."""
    if isinstance(expr, Or):
        and_args = [arg for arg in expr.args if isinstance(arg, And)]
        if and_args:
            first_and = and_args[0]
            rest = [arg for arg in expr.args if arg != first_and]
            return And(*[distribute_ands(Or(arg, *rest)) for arg in
first_and.args])
    elif isinstance(expr, And) or expr.is_Atom or expr.is_Not:
        return expr
    return expr.func(*[distribute_ands(arg) for arg in expr.args])

def to_cnf(expr):
    """Convert the given logical expression to CNF."""
    expr = eliminate_implications(expr)
    expr = push_negations(expr)
    expr = distribute_ands(expr)
    return expr

# Example usage:
A, B, C = symbols('A B C')
fol_expr = Implies(A, Or(B, Not(C)))  # Example FOL expression
cnf_expr = to_cnf(fol_expr)
print("FOL expression:", fol_expr)
print("CNF expression:", cnf_expr)
name = "Varsha Prasanth"
usn = "1BM22CS321"
print(f"Name: {name}, USN: {usn}")
```

OUTPUT



```
FOL expression: Implies(A, B | ~C)
CNF expression: B | ~A | ~C
Name: Varsha Prasanth, USN: 1BM22CS321
```