**LAB 7:**

**Program:** First Order Logic

**Code:**

import re

class FOLConverter:

    def \_\_init\_\_(self):

        self.constants = {}

        self.predicates = {}

    def define\_predicates(self, name, arity):

        self.predicates[name] = arity

    def define\_constants(self, name):

        self.constants[name] = name

    def translate(self, sentence):

        sentence = sentence.lower().strip()

        if self.match\_universal(sentence):

            return self.handle\_universal(sentence)

        elif self.match\_existential(sentence):

            return self.handle\_existential(sentence)

        elif self.match\_implication(sentence):

            return self.handle\_implication(sentence)

        else:

            return self.handle\_basic(sentence)

    def match\_universal(self, sentence):

        return re.match(r"(every|all)\s+\w+", sentence)

    def handle\_universal(self, sentence):

        match = re.match(r"(every|all)\s+(\w+)\s+is\s+(\w+)", sentence)

        if match:

            subject = match.group(2)

            predicate = match.group(3)

            return f"∀x ({subject}(x) → {predicate}(x))"

        return sentence

    def match\_existential(self, sentence):

        return re.match(r"there\s+is\s+(a|someone)\s+who", sentence)

    def handle\_existential(self, sentence):

        match = re.match(r"there\s+is\s+(a|someone)\s+who\s+(loves|knows)\s+(\w+)", sentence)

        if match:

            predicate = match.group(2)

            object = match.group(3)

            return f"∃x {predicate}(x, {object})"

        return sentence

    def match\_implication(self, sentence):

        return re.match(r"if\s+.\*\s+then\s+.\*", sentence)

    def handle\_implication(self, sentence):

        match = re.match(r"if\s+(.\*)\s+then\s+(.\*)", sentence)

        if match:

            premise = match.group(1)

            conclusion = match.group(2)

            return f"{premise} → {conclusion}"

        return sentence

    def handle\_basic(self, sentence):

        # For simplicity, we'll handle sentences like "John loves Mary" as a basic predicate

        match = re.match(r"(\w+)\s+is\s+(\w+)", sentence)

        if match:

            subject = match.group(1)

            predicate = match.group(2)

            return f"{predicate}({subject})"

        return sentence

fol\_converter = FOLConverter()

fol\_converter.define\_predicates("Human", 1)

fol\_converter.define\_predicates("Mortal", 1)

fol\_converter.define\_predicates("Loves", 2)

fol\_converter.define\_predicates("Student", 1)

sentences = [

    "John is a human",

    "Every human is mortal",

    "John loves Mary",

    "There is someone who loves Mary",

    "If it is raining, then the ground is wet"

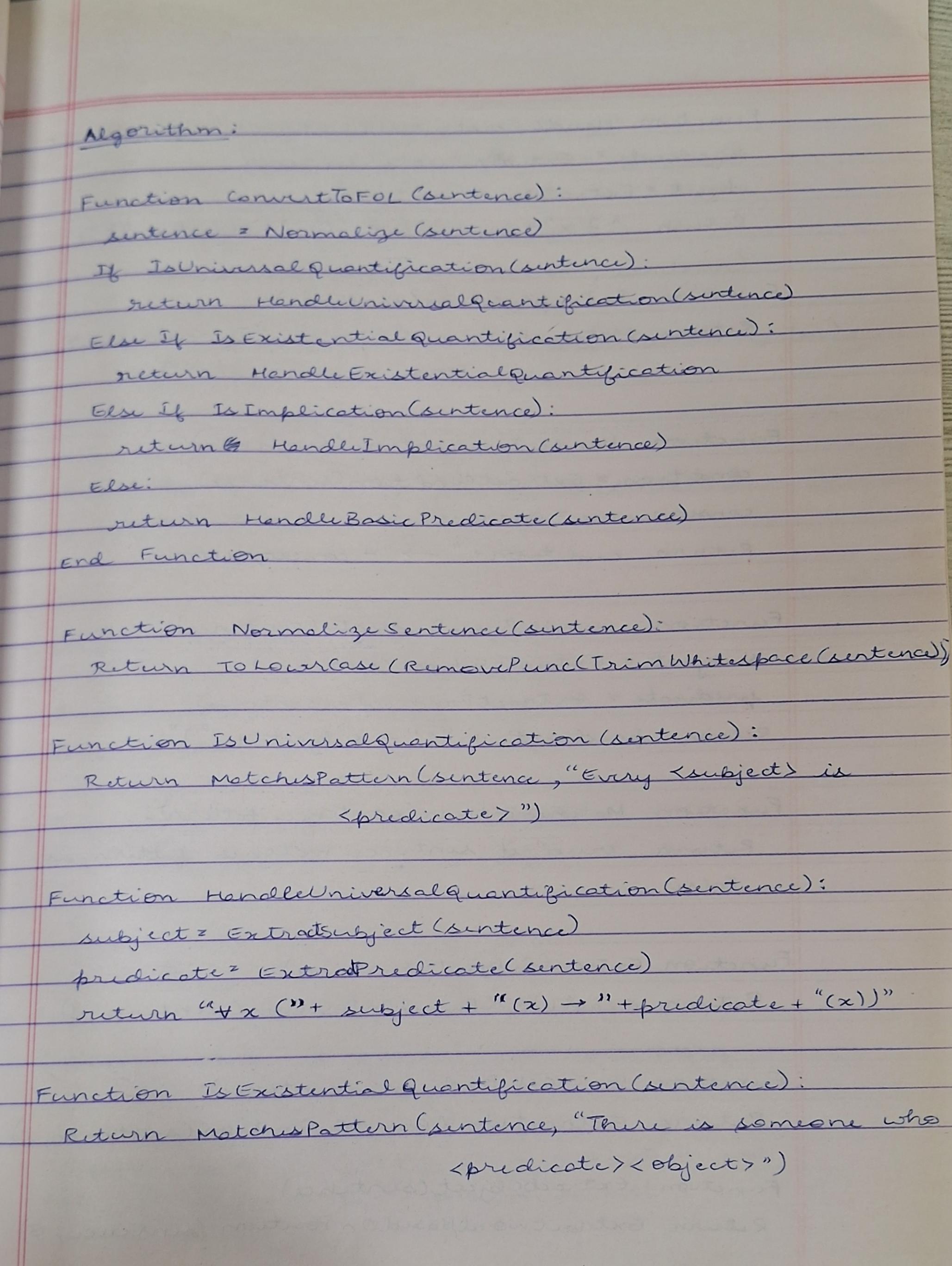
]

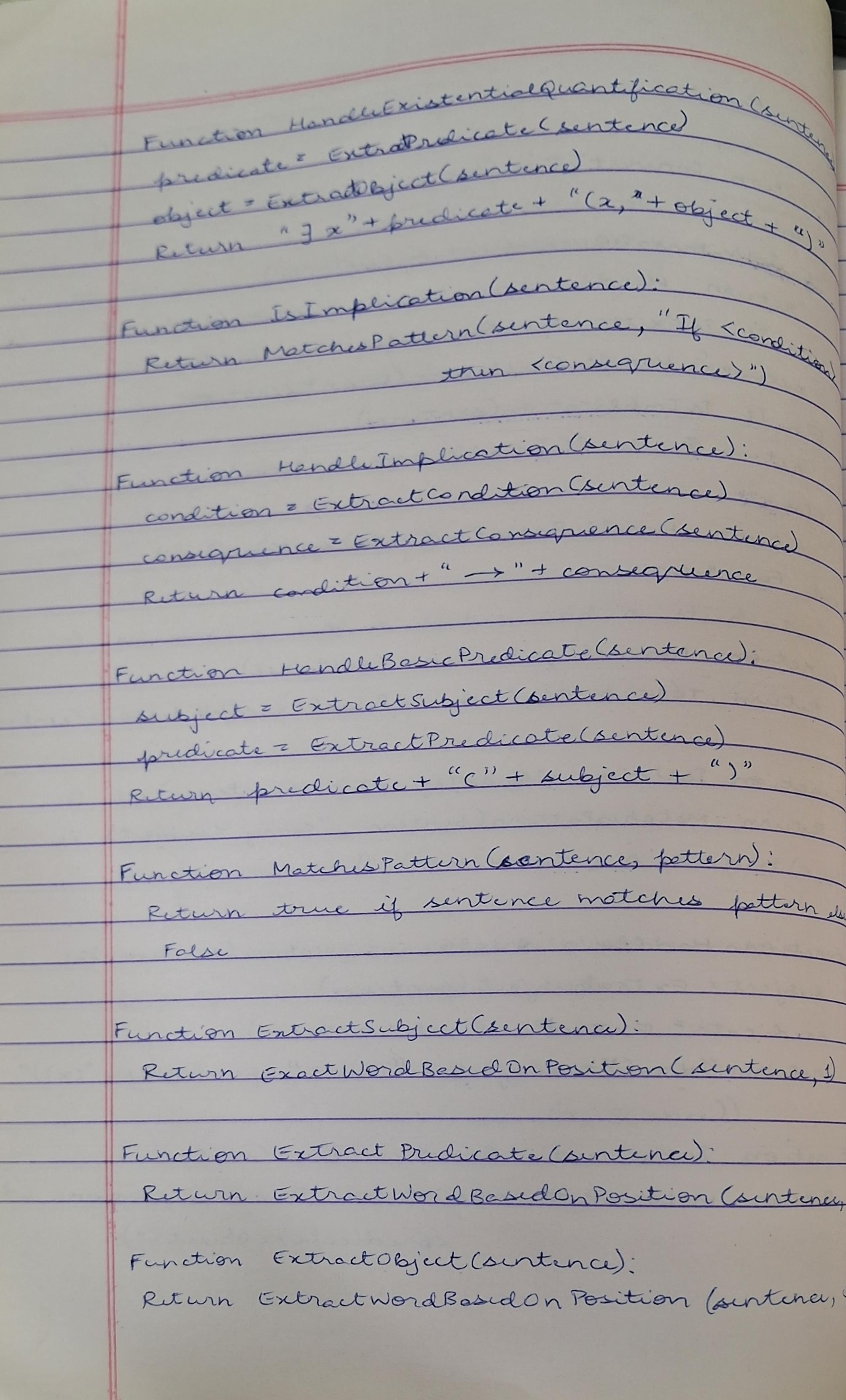
for sentence in sentences:

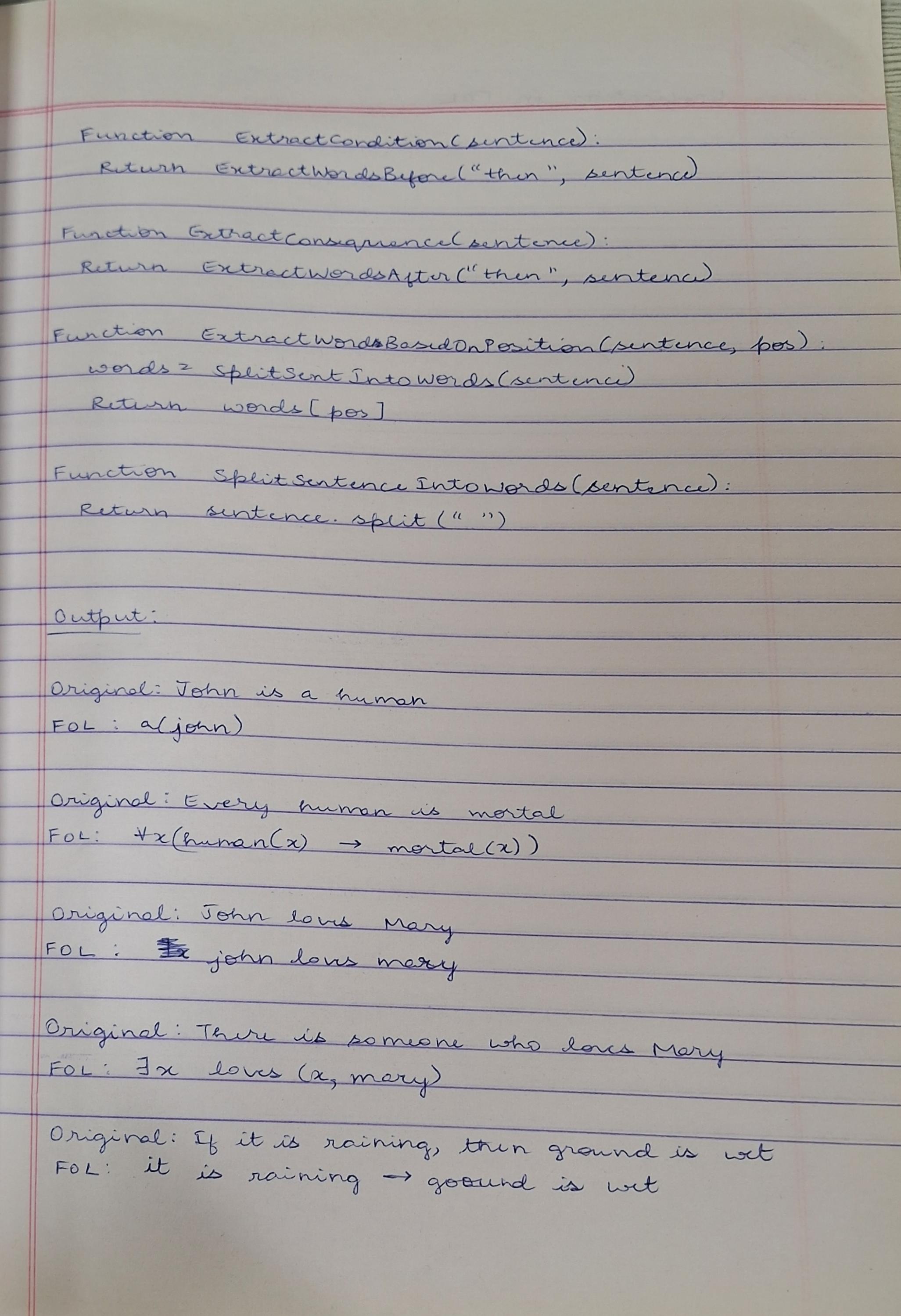
    fol = fol\_converter.translate(sentence)

    print(f"Original: {sentence}")

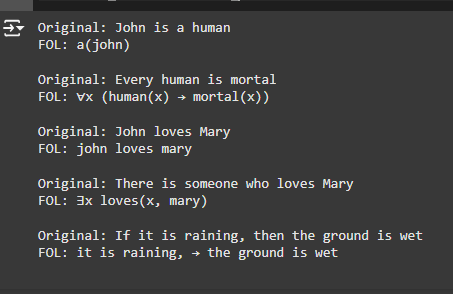
    print(f"FOL: {fol}\n")

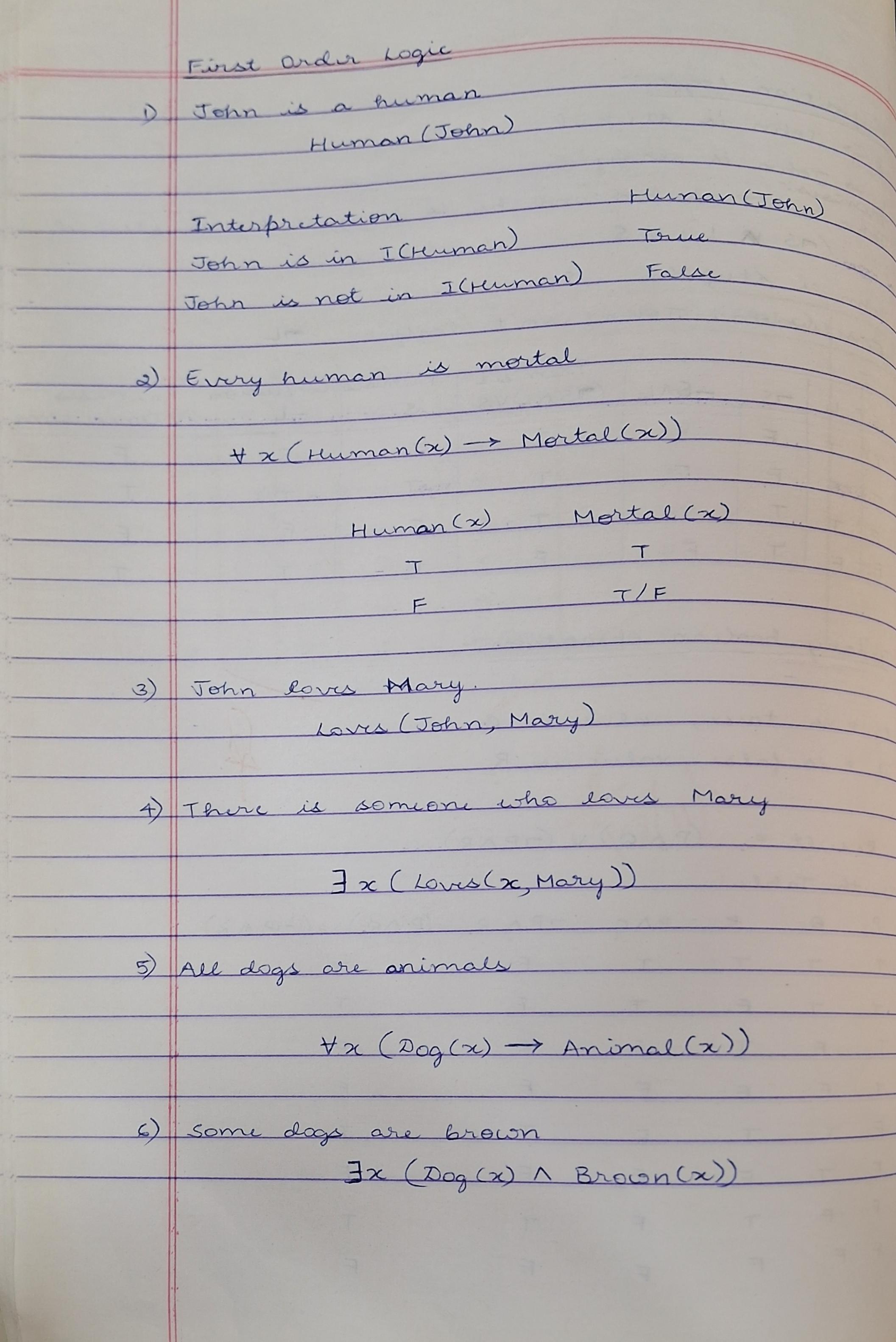
**Algorithm:**

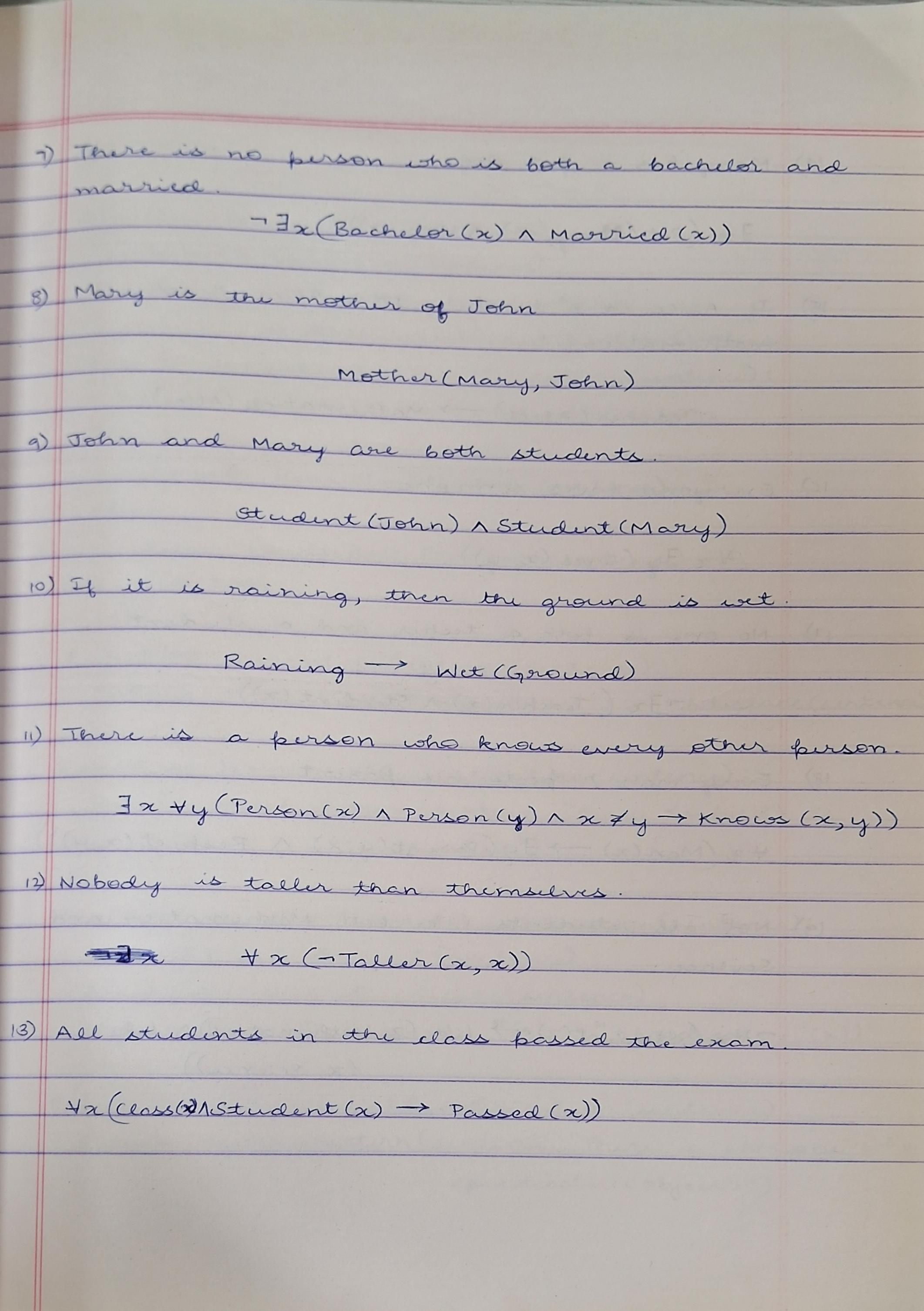
****

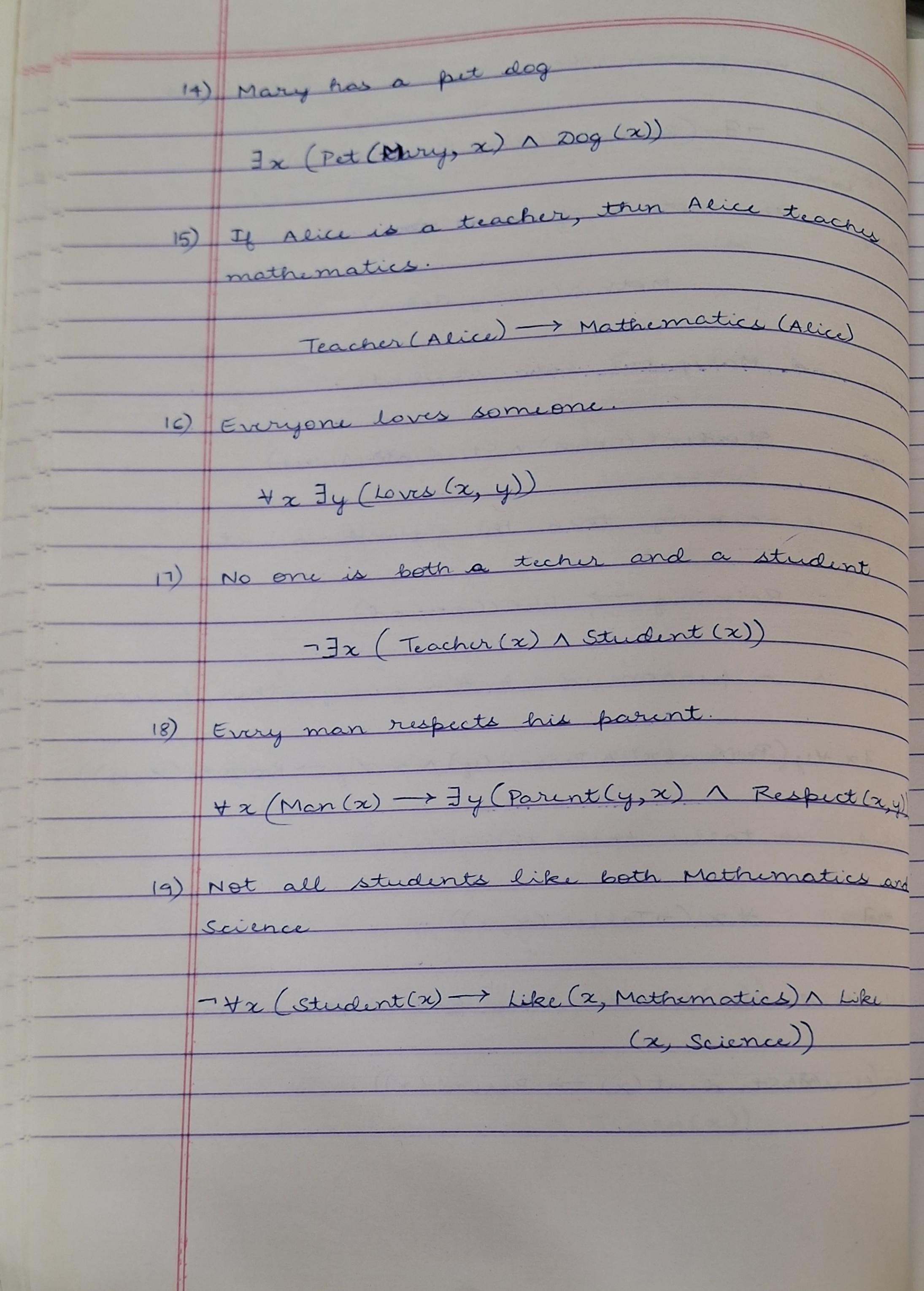
****

**Output:**

****

****

****

****