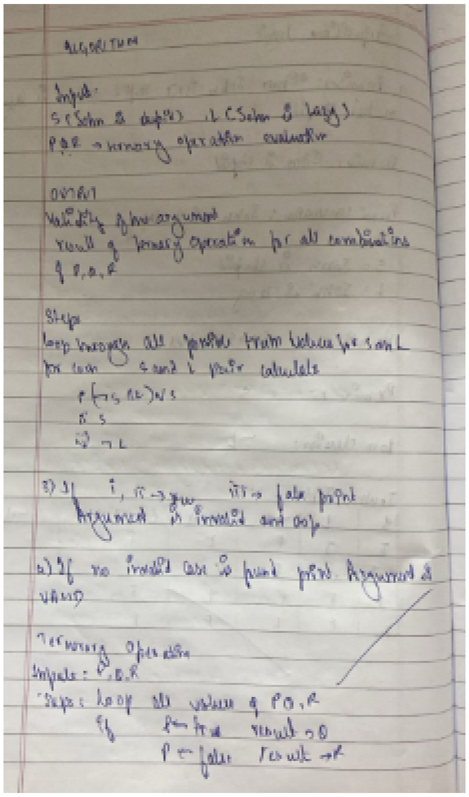
Date 19/11/24

Program Title: FOL and Prepositional Logic

Algorithm:

Prepositional Logic:



ALGORITHM

Input:
S(John is diligent) L(John is lazy)
P,Q,R → ternary operation evaluation

OUTPUT
Validity of the argument
result of ternary operation for all combinations of P,Q,R

Steps
loop through all possible truth values for S and L
for each S and L pair calculate

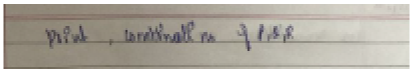
```
    i (¬S ∧ L) ∨ S
    ii S
    iii ¬L
```

3) If i, ii → true iii → false print
Argument is invalid and stop

4) If no invalid case is found print Argument is VALID

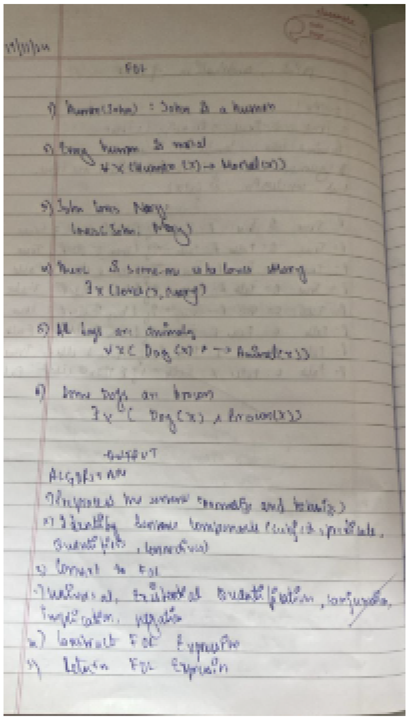
Ternary Operation
Inputs: P,Q,R
Steps: loop all values of P,Q,R

```
    if  P ← true  result → Q
        P ← false result → R
```



print combination of P,Q,R

FOL:



17/11/24

FOL

1) Human(John) : John is a human

2) Every human is mortal
   $\forall x (Human(x) \rightarrow Mortal(x))$

3) John loves Mary
   loves(John, Mary)

4) There is some-one who loves Mary
   $\exists x (loves(x, Mary))$

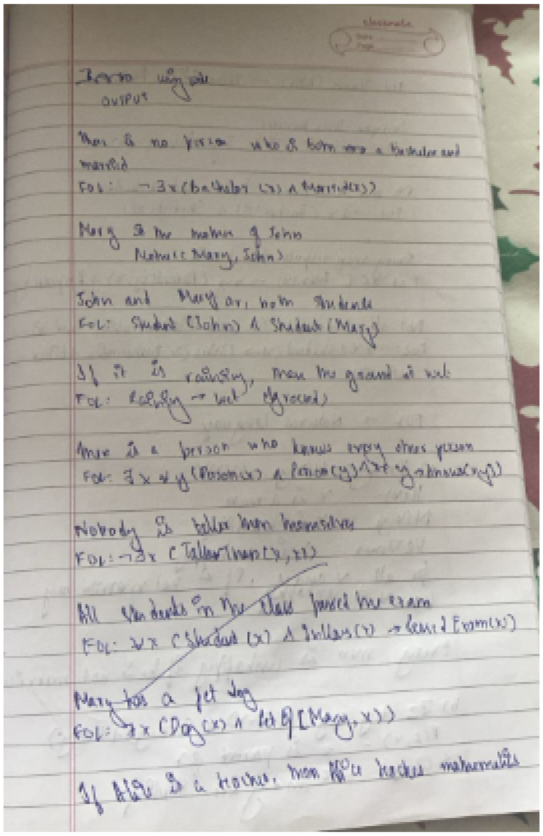
5) All dogs are animals
   $\forall x (Dog(x) \rightarrow Animal(x))$

6) Some dogs are brown
   $\exists x (Dog(x) \land Brown(x))$

OUTPUT

ALGORITHM

1) Preprocess the sentence (normalize and tokenize)
2) Identify sentence components (entities, predicates, quantifiers, connectives)
3) Convert to FOL
- Universal, existential quantification, conjunction, implication, negation
4) Construct FOL Expression
5) Return FOL Expression



[illegible]

OUTPUT

There is no person who is both a bachelor and married
FOL: $\neg \exists x (bachelor(x) \land Married(x))$

Mary is the mother of John
   Mother(Mary, John)

John and Mary are both students
FOL: $Student(John) \land Student(Mary)$

If it is raining, then the ground is wet
FOL: $Raining \rightarrow Wet(ground)$

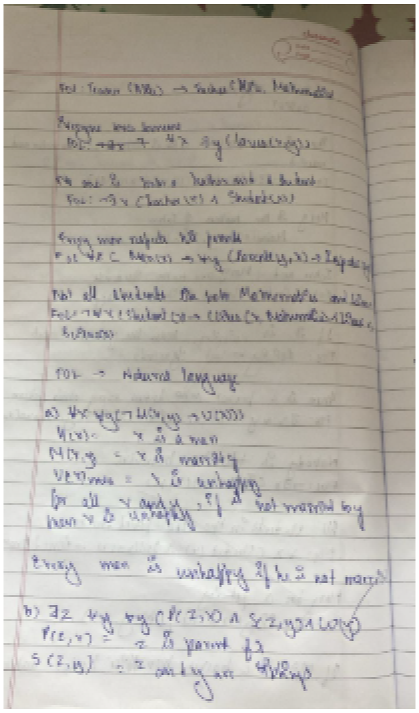
There is a person who knows every other person
FOL: $\exists x \forall y (Person(x) \land Person(y) \land x \neq y \rightarrow knows(x,y))$

Nobody is taller than themselves
FOL: $\neg \exists x (TallerThan(x,x))$

All students in the class passed the exam
FOL: $\forall x (Student(x) \land InClass(x) \rightarrow PassedExam(x))$

Mary has a pet dog
FOL: $\exists x (Dog(x) \land PetOf(Mary, x))$

If Alice is a teacher, then Alice teaches mathematics



FOL: $Teacher(Alice) \rightarrow Teaches(Alice, Mathematics)$

Everyone loves someone
FOL: ~~$\exists x$~~ $\forall x \exists y (loves(x,y))$

No one is both a bachelor and a student
FOL: $\neg \exists x (Bachelor(x) \land Student(x))$

Every man respects his parents
FOL: $\forall x (Man(x) \rightarrow \forall y (Parent(y,x) \rightarrow Respects(x,y)))$

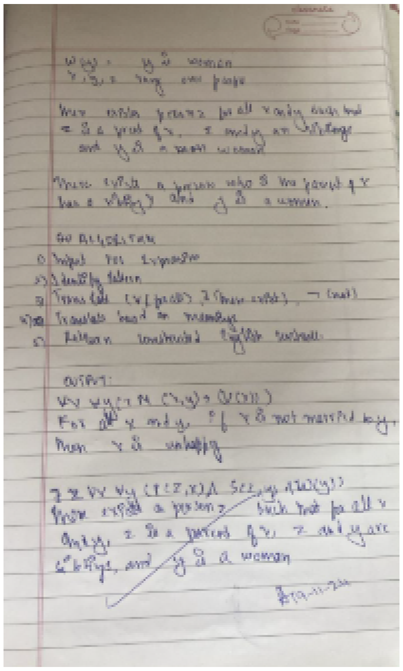
Not all students like both Mathematics and Science
FOL: $\neg \forall x (Student(x) \rightarrow (Likes(x, Mathematics) \land Likes(x, Science)))$

FOL → Natural language

a) $\forall x \forall y (\neg M(x,y) \rightarrow U(x))$
M(x) : x is a man
M(x,y) : x is married y
U(x) : x is unhappy
for all x and y, if x not married by y then x is unhappy

Every man is unhappy if he is not married

b) $\exists z \forall x \forall y (P(z,x) \land S(z,y) \land W(y))$
P(z,x) = z is parent of x
S(z,y) : z and y are siblings



W(y) : y is women
x, y, z : they are people

There exists person z for all x and y such that z is a parent of x, z and y are siblings and y is a women

There exists a person who is the parent of x has a sibling and y is a women.

ALGORITHM

1) Input FOL Expression
2) Identify pattern
3) Translate (∀ for all, ∃ there exists), ¬ (not)
4) Translate based on meaning
5) Return constructed English sentence

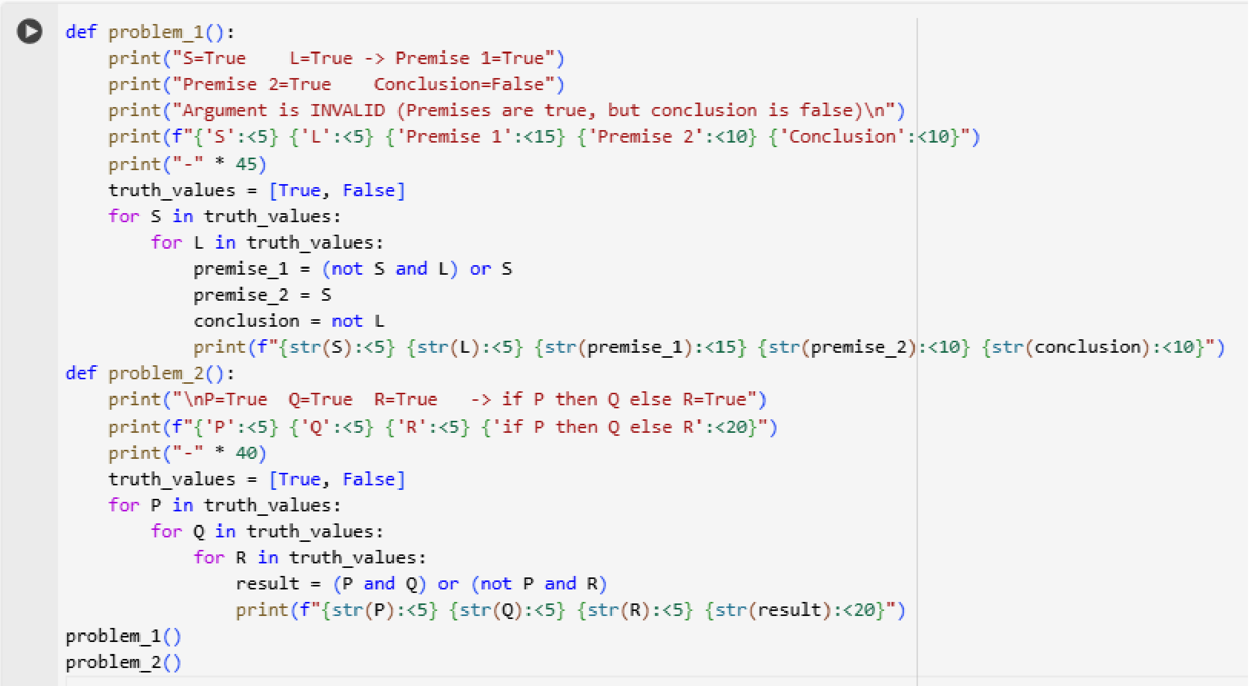
OUTPUT:
$\forall x \forall y (\neg M(x,y) \rightarrow U(x))$
For all x and y, if x is not married by y, then x is unhappy

$\exists z \forall x \forall y (P(z,x) \land S(z,y) \land W(y))$
There exists a person z such that for all x and y, z is a parent of x, z and y are siblings, and y is a woman

17-11-24

Code:

Preposition Logic:

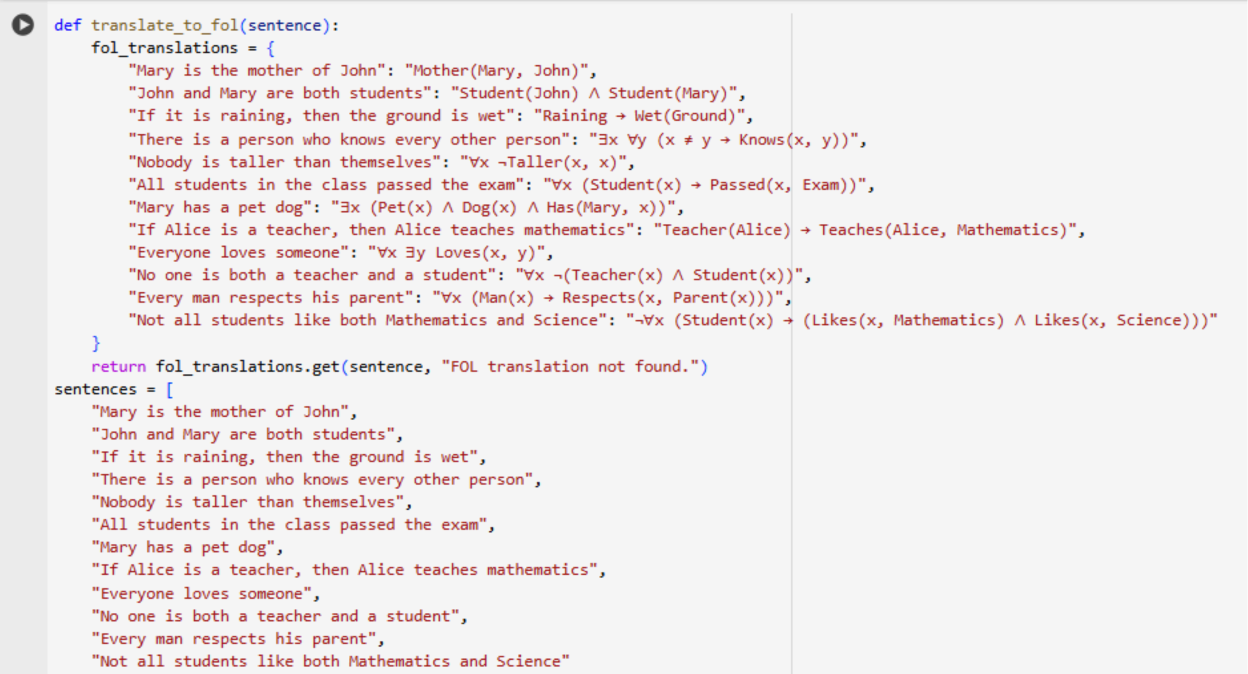


```
def problem_1():
    print("S=True    L=True -> Premise 1=True")
    print("Premise 2=True    Conclusion=False")
    print("Argument is INVALID (Premises are true, but conclusion is false)\n")
    print(f"{'S':<5} {'L':<5} {'Premise 1':<15} {'Premise 2':<10} {'Conclusion':<10}")
    print("-" * 45)
    truth_values = [True, False]
    for S in truth_values:
        for L in truth_values:
            premise_1 = (not S and L) or S
            premise_2 = S
            conclusion = not L
            print(f"{str(S):<5} {str(L):<5} {str(premise_1):<15} {str(premise_2):<10} {str(conclusion):<10}")
def problem_2():
    print("\nP=True  Q=True  R=True   -> if P then Q else R=True")
    print(f"{'P':<5} {'Q':<5} {'R':<5} {'if P then Q else R':<20}")
    print("-" * 40)
    truth_values = [True, False]
    for P in truth_values:
        for Q in truth_values:
            for R in truth_values:
                result = (P and Q) or (not P and R)
                print(f"{str(P):<5} {str(Q):<5} {str(R):<5} {str(result):<20}")
problem_1()
problem_2()
```

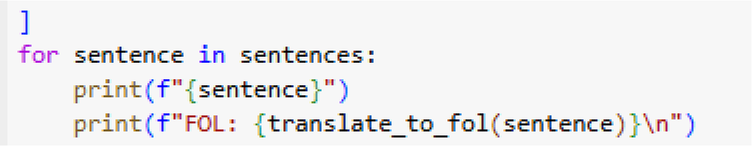
```
def problem_1():
    print("S=True    L=True -> Premise 1=True")
    print("Premise 2=True    Conclusion=False")
    print("Argument is INVALID (Premises are true, but conclusion is false)\n")
    print(f"{'S':<5} {'L':<5} {'Premise 1':<15} {'Premise 2':<10} {'Conclusion':<10}")
    print("-" * 45)
    truth_values = [True, False]
    for S in truth_values:
        for L in truth_values:
            premise_1 = (not S and L) or S
            premise_2 = S
            conclusion = not L
            print(f"{str(S):<5} {str(L):<5} {str(premise_1):<15} {str(premise_2):<10}
{str(conclusion):<10}")
def problem_2():
    print("\nP=True  Q=True  R=True   -> if P then Q else R=True")
    print(f"{'P':<5} {'Q':<5} {'R':<5} {'if P then Q else R':<20}")
    print("-" * 40)
    truth_values = [True, False]
    for P in truth_values:
        for Q in truth_values:
            for R in truth_values:
                result = (P and Q) or (not P and R)
                print(f"{str(P):<5} {str(Q):<5} {str(R):<5} {str(result):<20}")
problem_1()
```

problem_2()

FOL:

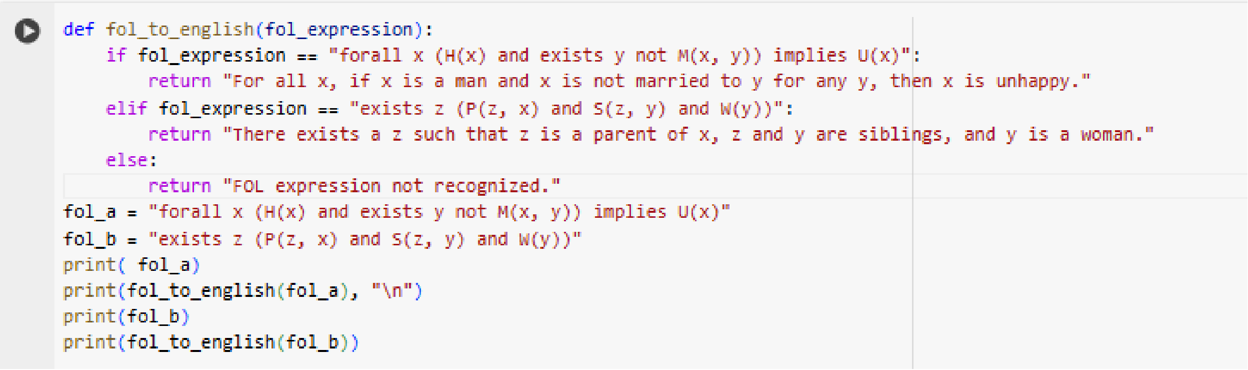


```
def translate_to_fol(sentence):
    fol_translations = {
        "Mary is the mother of John": "Mother(Mary, John)",
        "John and Mary are both students": "Student(John) ∧ Student(Mary)",
        "If it is raining, then the ground is wet": "Raining → Wet(Ground)",
        "There is a person who knows every other person": "∃x ∀y (x ≠ y → Knows(x, y))",
        "Nobody is taller than themselves": "∀x ¬Taller(x, x)",
        "All students in the class passed the exam": "∀x (Student(x) → Passed(x, Exam))",
        "Mary has a pet dog": "∃x (Pet(x) ∧ Dog(x) ∧ Has(Mary, x))",
        "If Alice is a teacher, then Alice teaches mathematics": "Teacher(Alice) → Teaches(Alice, Mathematics)",
        "Everyone loves someone": "∀x ∃y Loves(x, y)",
        "No one is both a teacher and a student": "∀x ¬(Teacher(x) ∧ Student(x))",
        "Every man respects his parent": "∀x (Man(x) → Respects(x, Parent(x)))",
        "Not all students like both Mathematics and Science": "¬∀x (Student(x) → (Likes(x, Mathematics) ∧ Likes(x, Science)))"
    }
    return fol_translations.get(sentence, "FOL translation not found.")
sentences = [
    "Mary is the mother of John",
    "John and Mary are both students",
    "If it is raining, then the ground is wet",
    "There is a person who knows every other person",
    "Nobody is taller than themselves",
    "All students in the class passed the exam",
    "Mary has a pet dog",
    "If Alice is a teacher, then Alice teaches mathematics",
    "Everyone loves someone",
    "No one is both a teacher and a student",
    "Every man respects his parent",
    "Not all students like both Mathematics and Science"
```



```
]
for sentence in sentences:
    print(f"{sentence}")
    print(f"FOL: {translate_to_fol(sentence)}\n")
```

```
def translate_to_fol(sentence):
    fol_translations = {
        "Mary is the mother of John": "Mother(Mary, John)",
        "John and Mary are both students": "Student(John) ∧ Student(Mary)",
        "If it is raining, then the ground is wet": "Raining → Wet(Ground)",
        "There is a person who knows every other person": "∃x ∀y (x ≠ y → Knows(x, y))",
        "Nobody is taller than themselves": "∀x ¬Taller(x, x)",
        "All students in the class passed the exam": "∀x (Student(x) → Passed(x, Exam))",
        "Mary has a pet dog": "∃x (Pet(x) ∧ Dog(x) ∧ Has(Mary, x))",
        "If Alice is a teacher, then Alice teaches mathematics": "Teacher(Alice) → Teaches(Alice,
Mathematics)",
        "Everyone loves someone": "∀x ∃y Loves(x, y)",
        "No one is both a teacher and a student": "∀x ¬(Teacher(x) ∧ Student(x))",
        "Every man respects his parent": "∀x (Man(x) → Respects(x, Parent(x)))",
        "Not all students like both Mathematics and Science": "¬∀x (Student(x) → (Likes(x,
Mathematics) ∧ Likes(x, Science)))"
    }
    return fol_translations.get(sentence, "FOL translation not found.")
sentences = [
    "Mary is the mother of John",
    "John and Mary are both students",
    "If it is raining, then the ground is wet",
    "There is a person who knows every other person",
    "Nobody is taller than themselves",
    "All students in the class passed the exam",
    "Mary has a pet dog",
    "If Alice is a teacher, then Alice teaches mathematics",
    "Everyone loves someone",
    "No one is both a teacher and a student",
    "Every man respects his parent",
    "Not all students like both Mathematics and Science"
]
for sentence in sentences:
    print(f"{sentence}")
    print(f"FOL: {translate_to_fol(sentence)}\n")
```

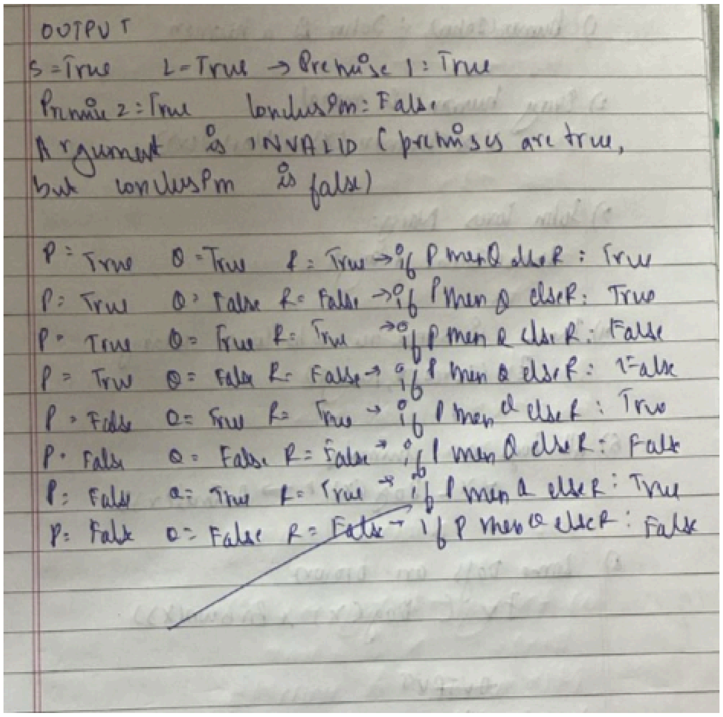


```
def fol_to_english(fol_expression):
    if fol_expression == "forall x (H(x) and exists y not M(x, y)) implies U(x)":
        return "For all x, if x is a man and x is not married to y for any y, then x is unhappy."
    elif fol_expression == "exists z (P(z, x) and S(z, y) and W(y))":
        return "There exists a z such that z is a parent of x, z and y are siblings, and y is a woman."
    else:
        return "FOL expression not recognized."
fol_a = "forall x (H(x) and exists y not M(x, y)) implies U(x)"
fol_b = "exists z (P(z, x) and S(z, y) and W(y))"
print( fol_a)
print(fol_to_english(fol_a), "\n")
print(fol_b)
print(fol_to_english(fol_b))
```

```
def fol_to_english(fol_expression):
    if fol_expression == "forall x (H(x) and exists y not M(x, y)) implies U(x)":
        return "For all x, if x is a man and x is not married to y for any y, then x is unhappy."
    elif fol_expression == "exists z (P(z, x) and S(z, y) and W(y))":
        return "There exists a z such that z is a parent of x, z and y are siblings, and y is a woman."
    else:
        return "FOL expression not recognized."
fol_a = "forall x (H(x) and exists y not M(x, y)) implies U(x)"
fol_b = "exists z (P(z, x) and S(z, y) and W(y))"
print( fol_a)
print(fol_to_english(fol_a), "\n")
print(fol_b)
print(fol_to_english(fol_b))
```

Snapshot of the output:

```
S=True    L=True -> Premise 1=True
Premise 2=True    Conclusion=False
Argument is INVALID (Premises are true, but conclusion is false)

S     L     Premise 1       Premise 2  Conclusion
---------------------------------------------
True  True  True            True       False
True  False True            True       True
False True  True            False      False
False False False           False      True

P=True  Q=True  R=True   -> if P then Q else R=True
P     Q     R     if P then Q else R
----------------------------------------
True  True  True  True
True  True  False True
True  False True  False
True  False False False
False True  True  True
False True  False False
False False True  True
False False False False
```



OUTPUT

S=True   L=True → Premise 1: True

Premise 2: True   Conclusion: False

Argument is INVALID (premises are true, but conclusion is false)

P: True Q: True R: True → if P then Q else R: True

P: True Q: False R: False → if P then Q else R: True

P: True Q: True R: True → if P then Q else R: False

P: True Q: False R: False → if P then Q else R: False

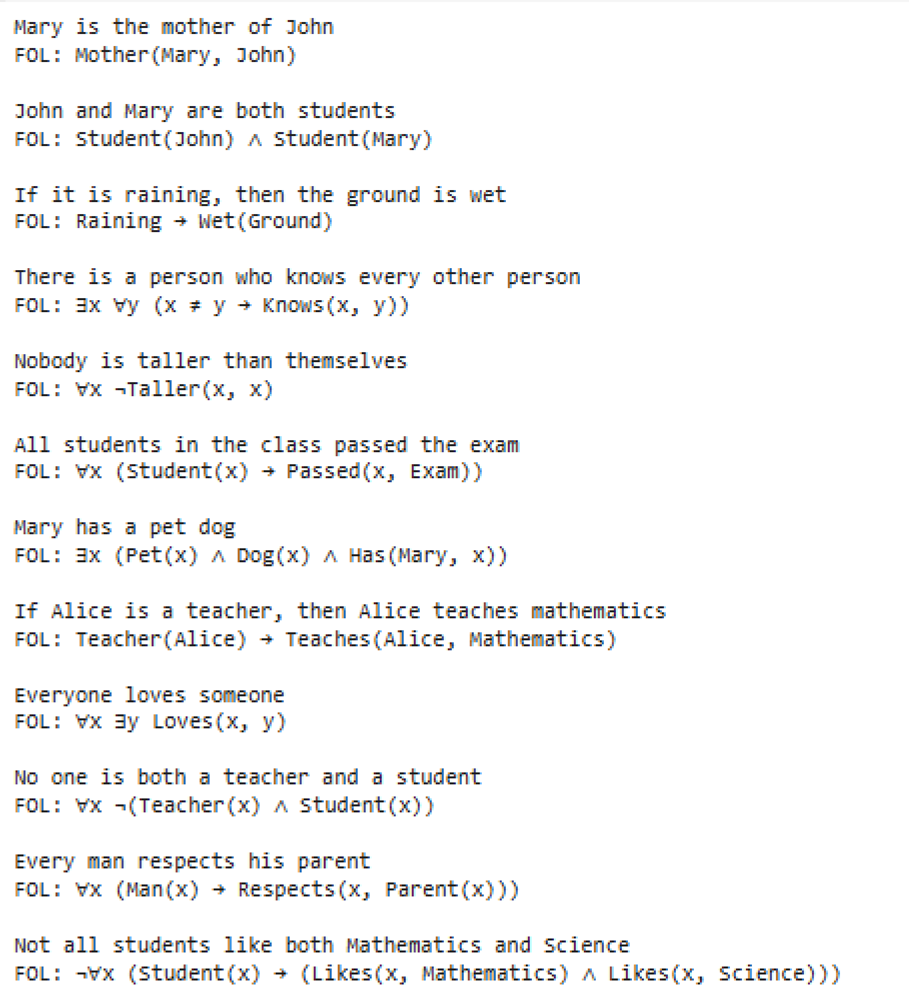
P: False Q: True R: True → if P then Q else R: True

P: False Q: False R: False → if P then Q else R: False

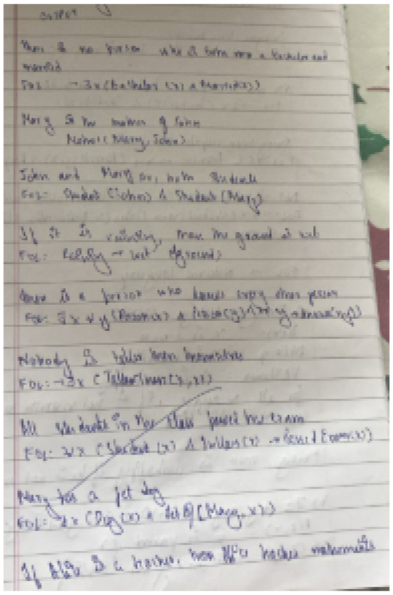
P: False Q: True R: True → if P then Q else R: True

P: False Q: False R: False → if P then Q else R: False

FOL:



```
Mary is the mother of John
FOL: Mother(Mary, John)

John and Mary are both students
FOL: Student(John) ∧ Student(Mary)

If it is raining, then the ground is wet
FOL: Raining → Wet(Ground)

There is a person who knows every other person
FOL: ∃x ∀y (x ≠ y → Knows(x, y))

Nobody is taller than themselves
FOL: ∀x ¬Taller(x, x)

All students in the class passed the exam
FOL: ∀x (Student(x) → Passed(x, Exam))

Mary has a pet dog
FOL: ∃x (Pet(x) ∧ Dog(x) ∧ Has(Mary, x))

If Alice is a teacher, then Alice teaches mathematics
FOL: Teacher(Alice) → Teaches(Alice, Mathematics)

Everyone loves someone
FOL: ∀x ∃y Loves(x, y)

No one is both a teacher and a student
FOL: ∀x ¬(Teacher(x) ∧ Student(x))

Every man respects his parent
FOL: ∀x (Man(x) → Respects(x, Parent(x)))

Not all students like both Mathematics and Science
FOL: ¬∀x (Student(x) → (Likes(x, Mathematics) ∧ Likes(x, Science)))
```



Output

There is no person who is both are a bachelor and married
FOL: ¬∃x (Bachelor(x) ∧ Married(x))

Mary is the mother of John
Mother(Mary, John)

John and Mary are both students
FOL: Student(John) ∧ Student(Mary)

If it is raining, then the ground is wet
FOL: Raining → Wet(Ground)

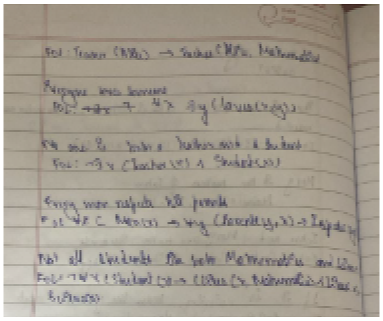
There is a person who knows every other person
FOL: ∃x ∀y (Person(x) ∧ Person(y) ∧ x≠y → Knows(x,y))

Nobody is taller than themselves
FOL: ¬∃x (Taller(x, x))

All students in the class passed the exam
FOL: ∀x (Student(x) ∧ InClass(x) → Passed(Exam(x)))

Mary has a pet dog
FOL: ∃x (Dog(x) ∧ Has(Mary, x))

If Alice is a teacher, then Alice teaches mathematics



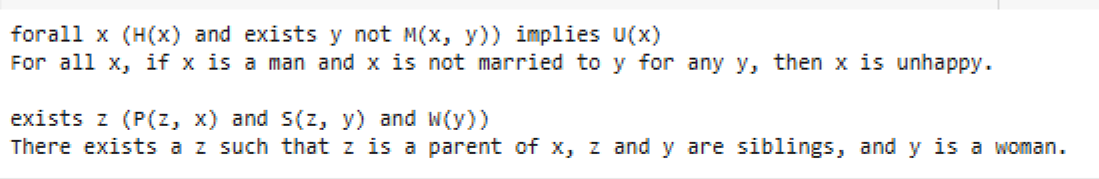
FOL: Teacher(Alice) → Teaches(Alice, Mathematics)

Everyone loves someone
FOL: ~~∀x~~ ∀x ∃y (Loves(x,y))

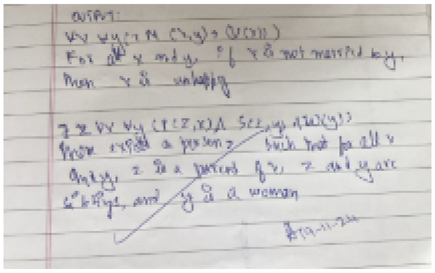
No one is both a teacher and a student
FOL: ¬∃x (Teacher(x) ∧ Student(x))

Every man respects his parents
FOL: ∀x (Man(x) → ∀y (Parent(y, x) → Respects(x, y

Not all students like both Mathematics and Science
FOL: ¬∀x (Student(x) → (Likes(x, Mathematics) ∧ Likes(x, Science)))



```
forall x (H(x) and exists y not M(x, y)) implies U(x)
For all x, if x is a man and x is not married to y for any y, then x is unhappy.

exists z (P(z, x) and S(z, y) and W(y))
There exists a z such that z is a parent of x, z and y are siblings, and y is a woman.
```



OUTPUT:

$\forall x\ \forall y(\neg M(x,y) \rightarrow U(x))$

For all x and y, if x is not married to y, then x is unhappy

$\exists z\ \forall x\ \forall y\ (P(z,x) \wedge S(z,y) \wedge W(y))$

There exists a person z such that for all x and y, z is a parent of x, z and y are siblings, and y is a woman

19-11-24