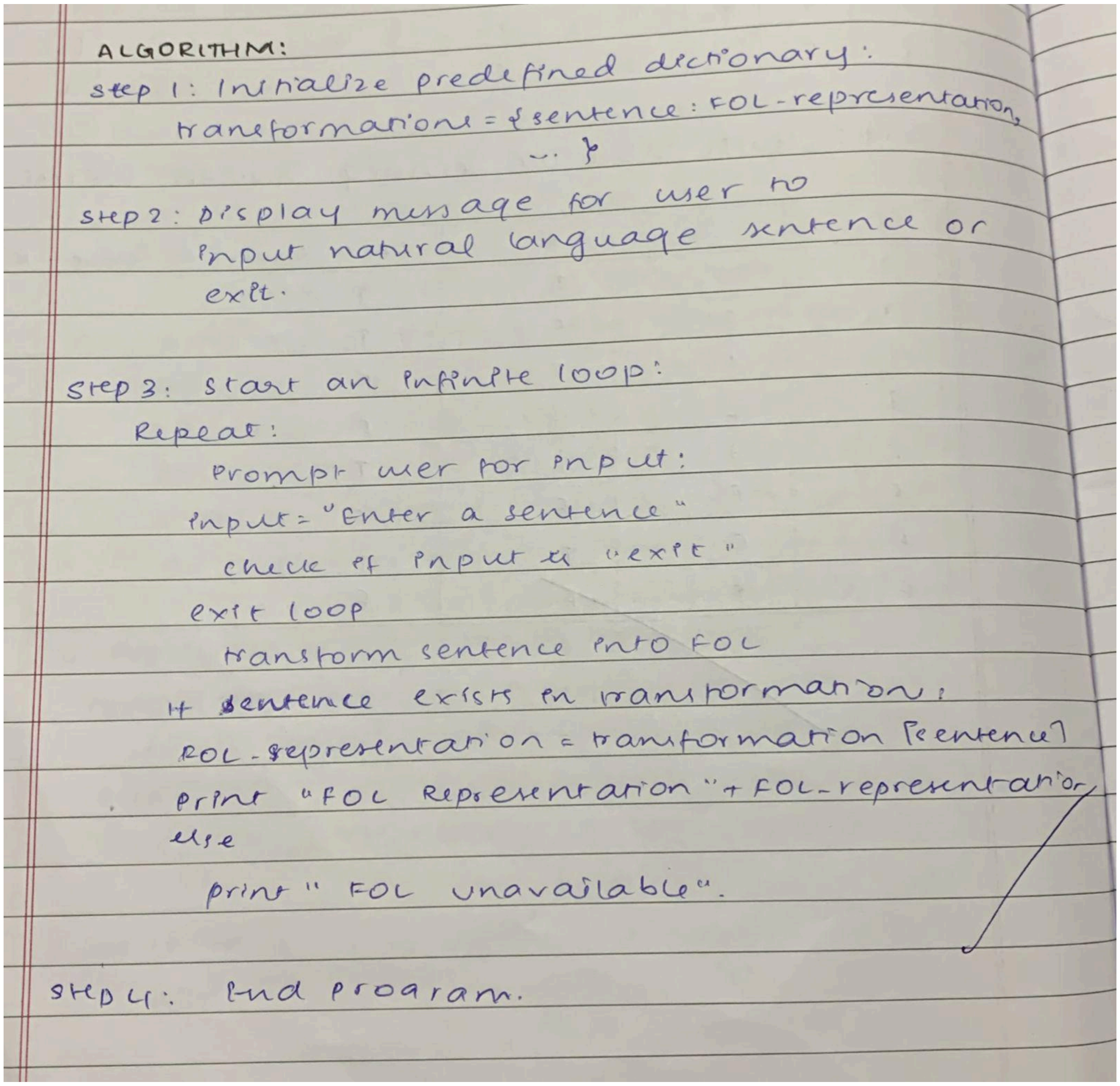
19-11-2024 WEEK - 7

First Order Logic

ALGORITHM-



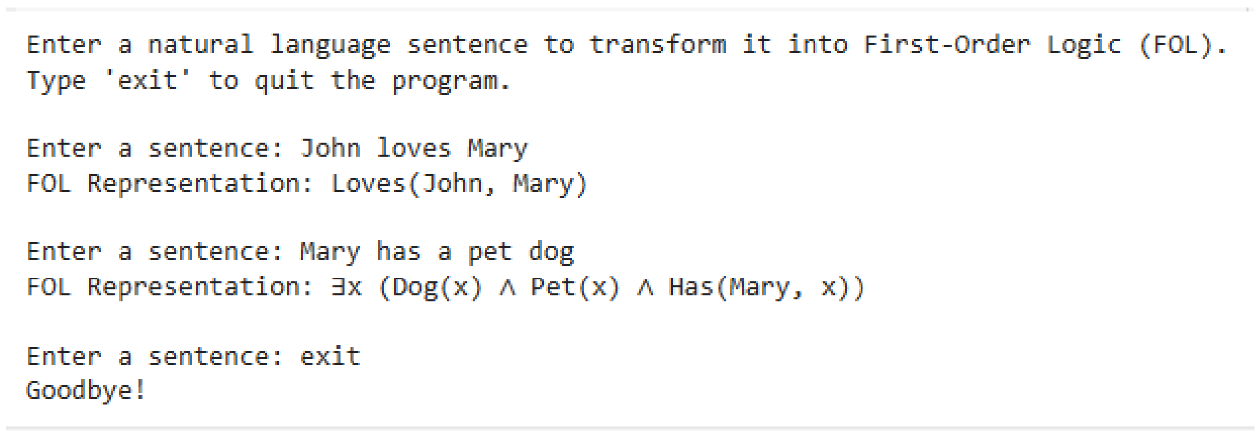
```
ALGORITHM:
step 1: Initialize predefined dictionary:
        transformations = {sentence: FOL-representation,
                           ... }

step 2: Display message for user to
        input natural language sentence or
        exit.

step 3: start an infinite loop:
    Repeat:
        Prompt user for input:
        input = "Enter a sentence"
        check if input is "exit"
        exit loop
        transform sentence into FOL
      if sentence exists in transformation:
        FOL-representation = transformation [sentence]
        print "FOL Representation" + FOL-representation
      else
        print " FOL unavailable".

step 4: End program.
```

CODE-

```
def transform_to_fol(sentence):
    transformations = {
        "John is a human": "Human(John)",
        "Every human is mortal": "∀x (Human(x) → Mortal(x))",
        "John loves Mary": "Loves(John, Mary)",
        "There is someone who loves Mary": "∃x (Loves(x, Mary))",
        "All dogs are animals": "∀x (Dog(x) → Animal(x))",
        "Some dogs are brown": "∃x (Dog(x) ∧ Brown(x))",
        "There is no person who is both a bachelor and married": "∀x (Person(x) → ¬(Bachelor(x)
∧ Married(x)))",
        "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 (Person(x) ∧ Person(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))",
        "Mary has a pet dog": "∃x (Dog(x) ∧ Pet(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 ∀y (Man(x) ∧ Parent(y, x) → Respects(x, y))",
        "Not all students like both Mathematics and Science": "¬∀x (Student(x) → (Likes(x, Math)
∧ Likes(x, Science)))",
    }
    return transformations.get(sentence, "FOL not available for this sentence")

def main():
    print("Enter a natural language sentence to transform it into First-Order Logic (FOL).")
    print("Type 'exit' to quit the program.\n")

    while True:
        sentence = input("Enter a sentence: ").strip()
        if sentence.lower() == "exit":
            print("Goodbye!")
            break
        fol = transform_to_fol(sentence)
        print(f"FOL Representation: {fol}\n")
```

```
if __name__ == "__main__":
    main()
```

OUTPUT-



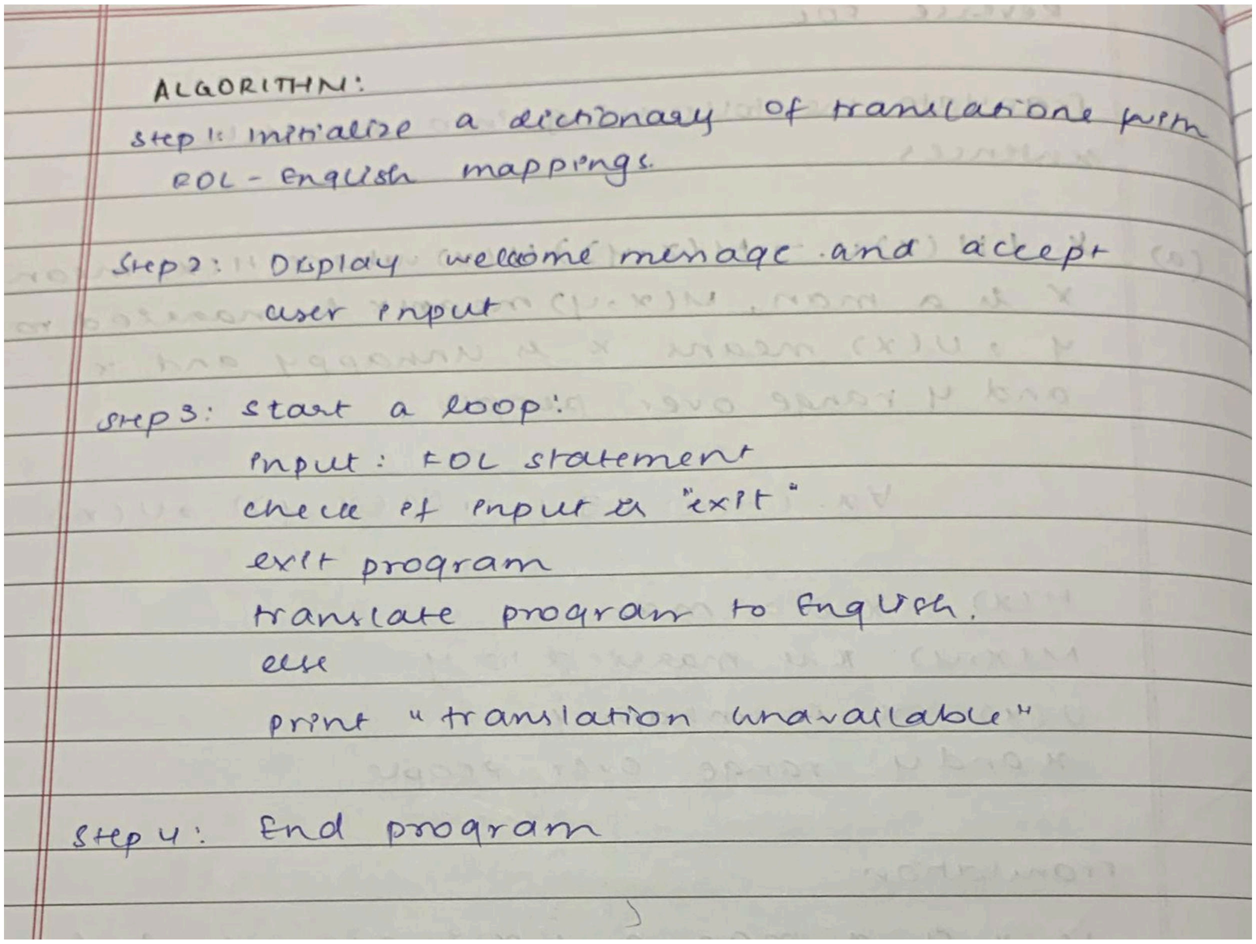
```
Enter a natural language sentence to transform it into First-Order Logic (FOL).
Type 'exit' to quit the program.

Enter a sentence: John loves Mary
FOL Representation: Loves(John, Mary)

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

Enter a sentence: exit
Goodbye!
```

Translate the following into English sentences:

(a) x.(H(x) y.¬M(x,y)) U(x) where H(x) means x is a man, M(x,y) means x is married to y, U(x) means x
is unhappy, and x and y range over people.

(b) z.P(z,x) S(z,y) W(y) where P(z,x) means z is a parent of x, S(z,y) means z and y are siblings, W(y)
means y is a woman, and x, y, and z range over people.

ALGORITHM-

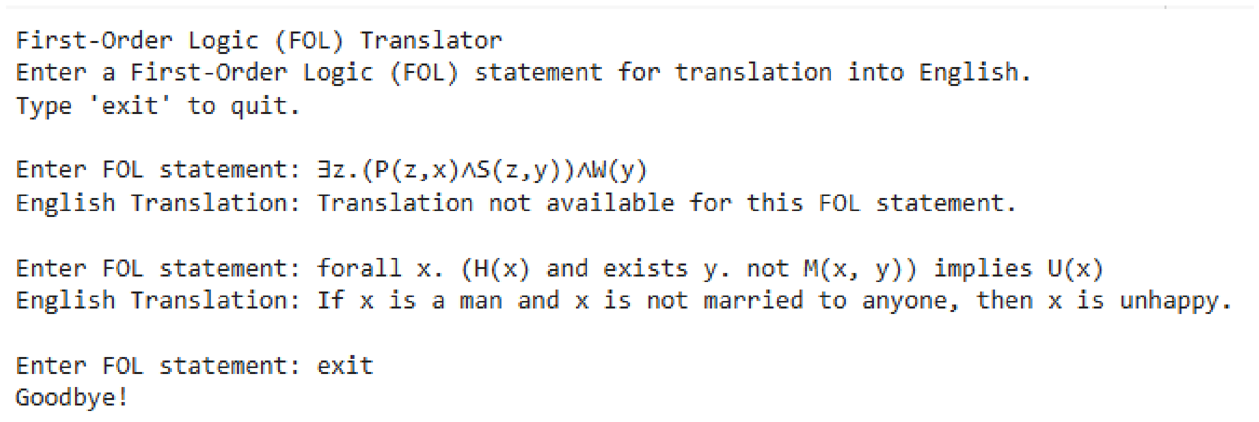


```
ALGORITHM:
Step 1: Initialise a dictionary of translations with
   FOL - English mappings.

Step 2: Display welcome message and accept
   user input

Step 3: Start a loop:
        Input: FOL statement
        check if input is "exit"
        exit program
        translate program to English.
        else
         print "translation unavailable"

Step 4: End program
```

CODE-

```
def translate_fol(statement):
    translations = {
        "forall x. (H(x) and exists y. not M(x, y)) implies U(x)":
            "If x is a man and x is not married to anyone, then x is unhappy.",

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

    return translations.get(statement, "Translation not available for this FOL statement.")
def main():
    print("First-Order Logic (FOL) Translator")
    print("Enter a First-Order Logic (FOL) statement for translation into English.")
    print("Type 'exit' to quit.\n")

    while True:
        statement = input("Enter FOL statement: ").strip()
        if statement.lower() == "exit":
            print("Goodbye!")
            break
        translation = translate_fol(statement)
        print(f"English Translation: {translation}\n")


if __name__ == "__main__":
    main()
```

OUTPUT-



```
First-Order Logic (FOL) Translator
Enter a First-Order Logic (FOL) statement for translation into English.
Type 'exit' to quit.

Enter FOL statement: ∃z.(P(z,x)∧S(z,y))∧W(y)
English Translation: Translation not available for this FOL statement.

Enter FOL statement: forall x. (H(x) and exists y. not M(x, y)) implies U(x)
English Translation: If x is a man and x is not married to anyone, then x is unhappy.

Enter FOL statement: exit
Goodbye!
```