

# **Programming Languages**

## **Homework 5 (Midterm)**

### **Report**

**Mustafa Tokgöz**  
**171044077**

# Explanation of code:

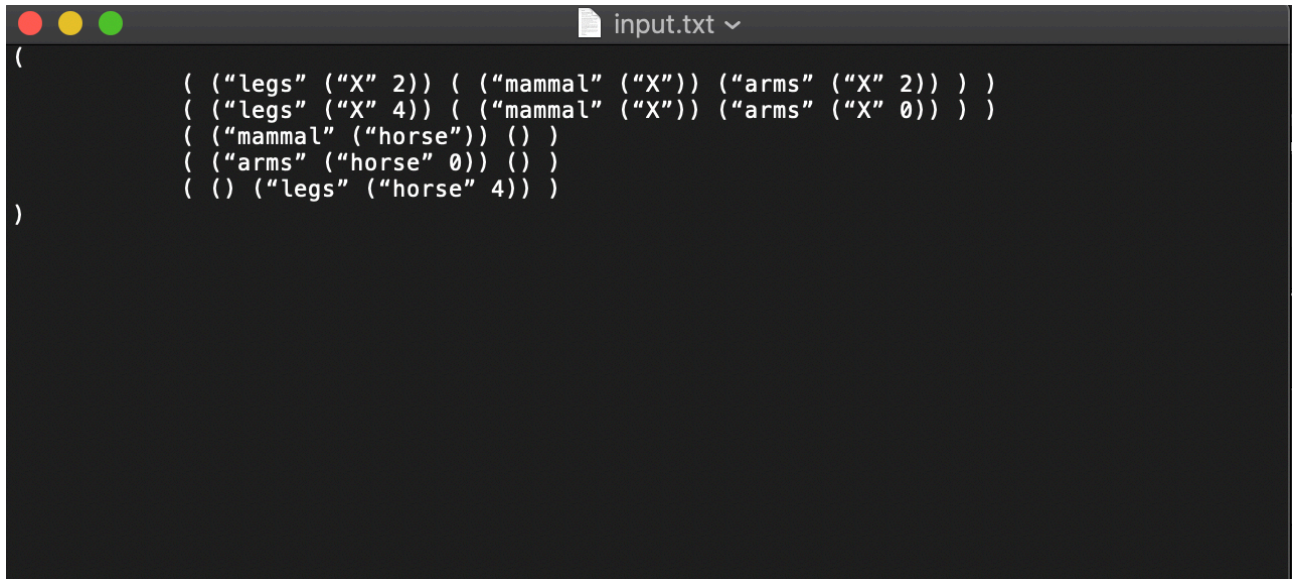
In this homework, First I read the input from input.txt as a list of sentence. After that I convert this sentences to lists. After that I take this lists and make resolution them by creating facts. I create facts by looking the second element of input list is null or not. Then if it is null , I add this fact first element to factlist list and the second list to factlistr list. After that I check the predicates by looking that there are not any null element. After that I add this predicate first element to predicatedlist list and second element to predicatedlistr list. Why I do this, because when I make unification, I use this informations to solve the input. After that I call unification part. After that I check the input is query is not by looking the first element of this sentence is null or not. After that I check the rest of query is equal to in factlist and factlistr by looping inside. After that I check for the predicate by changing parameter of query with predicate parameter and I change also inside of predicate with this parameter. After that if inside of the predicate facts is returns 1 then predicate returns 1. Then I write true to output.txt file if it is 1. Also If the query is fact then it returns 1. Then I write true to output.txt file if it is true. If it is not 1 then I write empty (nil) to output.txt file. This program works fine.

Implemented properties : Query, fact, predicate with resolution and unification method.

How to handle them is explained upper side.

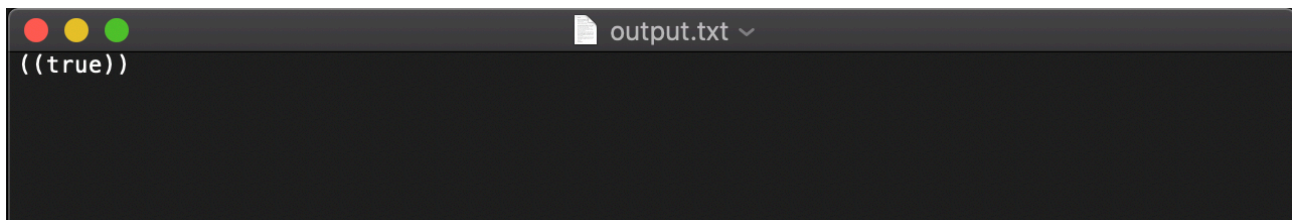
## Example

### Input File:



```
(
    ( ("legs" ("X" 2)) ( ("mammal" ("X")) ("arms" ("X" 2)) ) )
    ( ("legs" ("X" 4)) ( ("mammal" ("X")) ("arms" ("X" 0)) ) )
    ( ("mammal" ("horse")) ( ) )
    ( ("arms" ("horse" 0)) ( ) )
    ( ( ) ("legs" ("horse" 4)) )
)
```

### Output File:



```
((true))
```

### Explanation:

true -> ( ( ) ("legs" ("horse" 4)) )

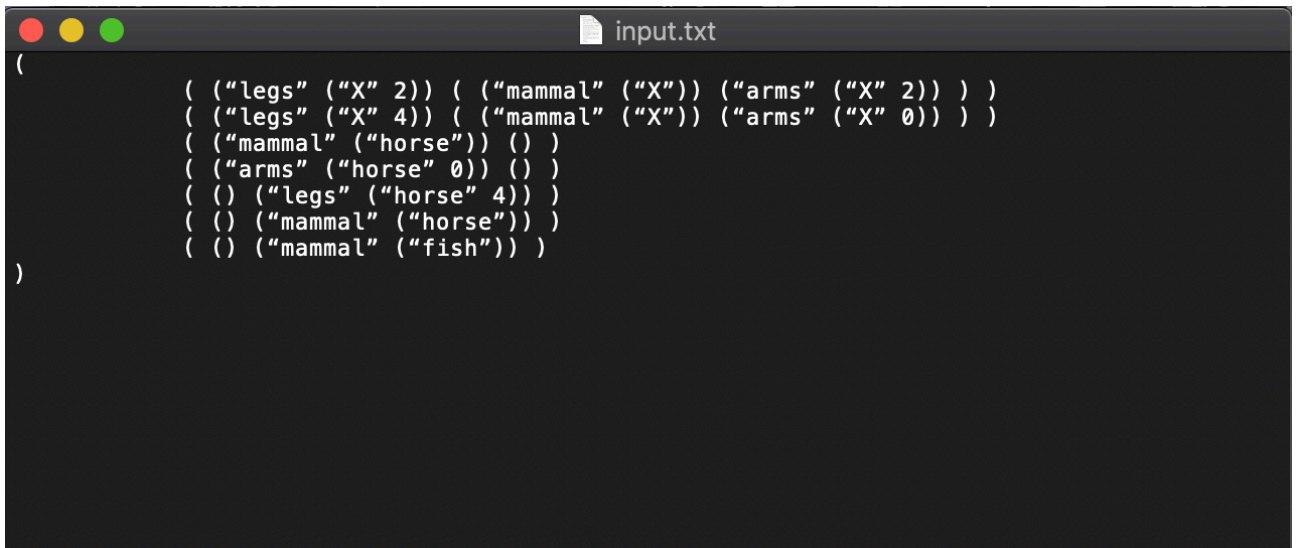
because mammal(horse) and arms(horse,0) are true

( ("mammal" ("horse")) ( ) )

( ("arms" ("horse" 0)) ( ) )

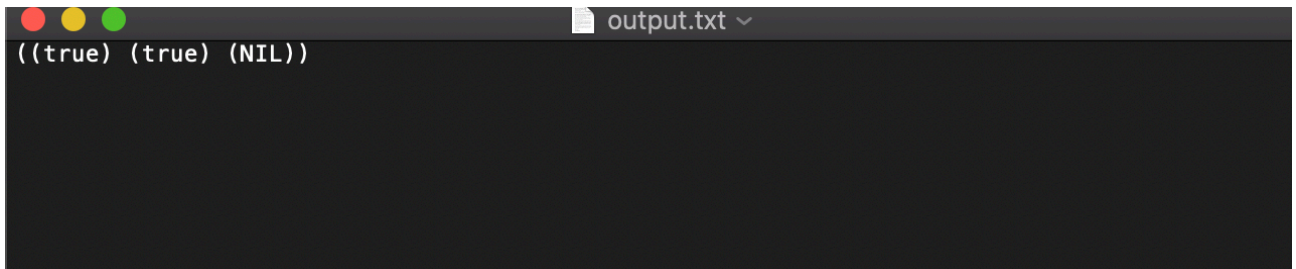
## Example

Input file :



```
(
  ( ("legs" ("X" 2)) ( ("mammal" ("X")) ("arms" ("X" 2)) ) )
  ( ("legs" ("X" 4)) ( ("mammal" ("X")) ("arms" ("X" 0)) ) )
  ( ("mammal" ("horse")) () )
  ( ("arms" ("horse" 0)) () )
  ( () ("legs" ("horse" 4)) )
  ( () ("mammal" ("horse")) )
  ( () ("mammal" ("fish")) )
)
```

Output file:



```
((true) (true) (NIL))
```

Explanation:

first true -> ( () ("legs" ("horse" 4)) )

because mammal(horse) and arms(horse,0) are true

( ("mammal" ("horse")) () )

( ("arms" ("horse" 0)) () )

second true -> ( () ("mammal" ("horse"))) )

because mammal(horse) is true

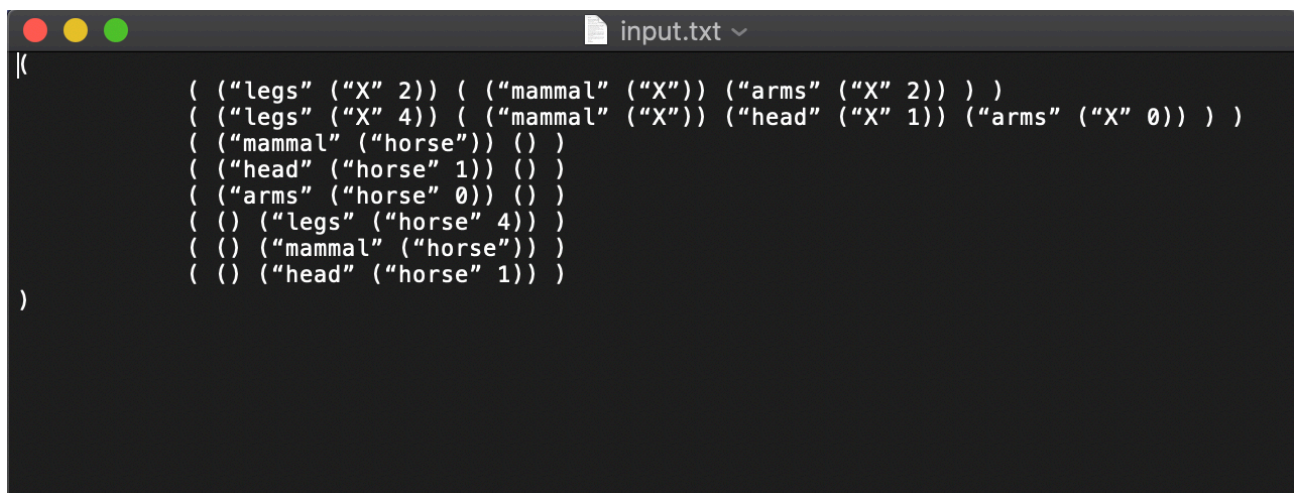
( ("mammal" ("horse")) () )

empty (nil) -> ( () ("mammal" ("fish"))) )

because mammal(fish) is false.

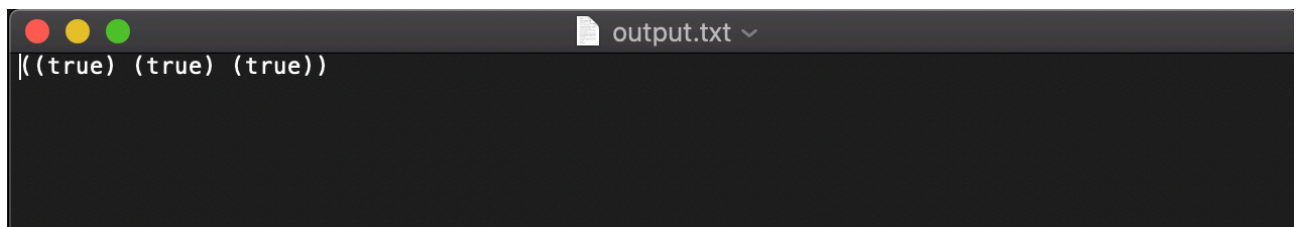
## Example

### Input File :



```
( ( ("legs" ("X" 2)) ( ("mammal" ("X")) ("arms" ("X" 2)) ) ) )  
  ( ("legs" ("X" 4)) ( ("mammal" ("X")) ("head" ("X" 1)) ("arms" ("X" 0)) ) )  
  ( ("mammal" ("horse")) () )  
  ( ("head" ("horse" 1)) () )  
  ( ("arms" ("horse" 0)) () )  
  ( () ("legs" ("horse" 4)) )  
  ( () ("mammal" ("horse")) )  
  ( () ("head" ("horse" 1)) )  
)
```

### Output File:



```
((true) (true) (true))
```

## Explanation:

first true -> ( () ("legs" ("horse" 4)) )

because mammal(horse) and head(horse,1) and arms(horse,0) are true

( ("mammal" ("horse")) () )

( ("head" ("horse" 1)) () )

( ("arms" ("horse" 0)) () )

second true -> ( () ("mammal" ("horse")) )

because mammal(horse) is true.

( ("mammal" ("horse")) () )

third true -> ( () ("head" ("horse" 1)) )

because head(horse,1) is true

( ("head" ("horse" 1)) () )