Predicate Logic:-

e.g: $\forall n \in \mathbb{N}\ \exists n\ :\ isEven(n) \longleftrightarrow isEven(n^2)$

* Predicate is a function from a domain of disourse to a set of propositions

Quantifiers:-

① $\dfrac{\dfrac{\forall x\ P(x)}{c \text{ is in domain}}}{P(c)}$	Universal Instantiation	e.g: $\dfrac{\dfrac{\text{All men are mortal}}{\text{Socrates is a man}}}{\text{Socrates is mortal}}$

① $\exists x\ P(x)$ is a proposition which is true if $P(x)$ is true for atleast one element in D.o.D

② $\forall x\ P(x)$ is a proposition which is true if $P(x)$ is true for every elements in D.o.D

e.g: ① Square of an even number is even.

$$\forall x\ (E(x) \rightarrow E(x^2))$$

② Goldbach's Conjecture.

Every even number larger than 2 can be expressed as sum of two prime numbers

$$\forall x\ \Big[\big((x>2) \wedge E(x)\big) \rightarrow \exists a \exists b\ \big(P(a) \wedge P(b) \wedge (x = a+b)\big)\Big]$$

③ There are infinite primes

$$\forall x\ \exists y\ \big(P(y) \wedge (y > x)\big)$$

④ Bezout's Identity.

$$D = \mathbb{Z}$$

$$\forall a \forall b\ \exists \alpha \exists \beta\ (\gcd(a,b) = \alpha a + \beta b)$$

⑤ Division Rule.

$\boxed{\exists!\ (\text{exists uniquely})}$

$$\forall a \forall d\ \exists q \exists r \Big[\big((a = dq + r) \wedge (0 \leq r \leq d-1)\big) \wedge \big(\forall q' \forall r' \big(\big((a = q'd + r') \wedge (0 \leq r' \leq d-1)\big) \rightarrow \big((q' = q) \wedge (r' = r)\big)\big)\Big]$$

⑥ Prime factorization.

$$D = \mathbb{N}$$

$$PP(x) = \forall k \forall p\ \big((x = p^k) \wedge (P(p))\big)$$

$$\forall x\ \exists S \Big[\big(\forall y\ (y \in S \rightarrow PP(y)\big) \wedge \big(x = \prod_{y \in S} y\big)\Big]$$

⑦ Negate Goldbach's Conjecture.

$$\exists x \Big[\big((x > 2) \wedge E(x)\big) \vee \big(\forall a \forall b \big((P(a) \wedge P(b)) \rightarrow (x \neq a + b)\big)\big)\Big]$$

Nested Quantifiers :-

Let $D = \{1, 2, 3, 4\}$ and P be a binary predicate over D.

$a = \forall x \forall y\ P(x,y)$ $\quad$ $a' = \forall y \forall x\ P(x,y)$

$b = \forall x \exists y\ P(x,y)$ $\quad$ $b' = \exists y \forall x\ P(x,y)$

$c = \exists x \forall y\ P(x,y)$ $\quad$ $c' = \forall y \exists x\ P(x,y)$

$d = \exists x \exists y\ P(x,y)$ $\quad$ $d' = \exists y \exists x\ P(x,y)$

$$\begin{tikzcd} & a \equiv a' \arrow[dl] \arrow[dr] & \\ b' \arrow[d] & & c \arrow[d] \\ b \arrow[dr] & & c' \arrow[dl] \\ & d \equiv d' & \end{tikzcd}$$

These relations are always true.

a, a'

y \ x	1	2	3	4
1	T	T	T	T
2	T	T	T	T
3	T	T	T	T
4	T	T	T	T

b

y \ x	1	2	3	4
1		T		T
2	T		T	
3				
4				

b'

y \ x	1	2	3	4
1				
2				
3	T	T	T	T
4				

c

y \ x	1	2	3	4
1		T		
2		T		
3		T		
4		T		

c'

y \ x	1	2	3	4
1			T	
2		T		
3			T	
4		T		

d, d'

y \ x	1	2	3	4
1				
2		T		
3				
4				

Program for a:

```
def AxAyP():
    for x in D:
        for y in D:
            if not P(x,y):
                return false
    return true
```

Program for b:

```
def AxEyP():
    for x in D:
        z = false
        for y in D:
            if P(x,y):
                z = true
                break
        if not z:
```

```
    if not z:
        return false
    return z
```

Program for c:

```
def ExAyP():
    for x in D:
        for y in D:
            if not P(x,y):
                return false
        return true
    return false
```

Program for d:

```
def ExEyP():
    for x in D:
        for y in D:
            if P(x,y):
                return true
    return false.
```

Program for c':

```
def AyExP():
    for y in D:
        z = false
        for x in D:
            if P(x,y):
                z = true
                break
        if not z:
            return false.
    return z
```

Program for b':

```
def EyAxP():
    for y in D:
        for x in D:
            if not P(x,y):
                return false
        return true
    return false
```

$\exists x \forall y \exists z P(x,y,z) \rightarrow \begin{cases} P(x,y,z) \\ Q(x,y) = \exists z P(x,y,z) \\ R(x) = \forall y Q(x,y) \\ \emptyset = \exists x R(x) \end{cases}$

```
def ExAyEzP():
    for x in D:
        for y in D:
            for z in D:
                P = P(x,y,z)
```

```
def Q(x,y):
    for z in D:
        if P(x,y,z):
            return true
    return false
```

```
def Ø():
    for x in D:
        if R(x):
```

```
        return false
```

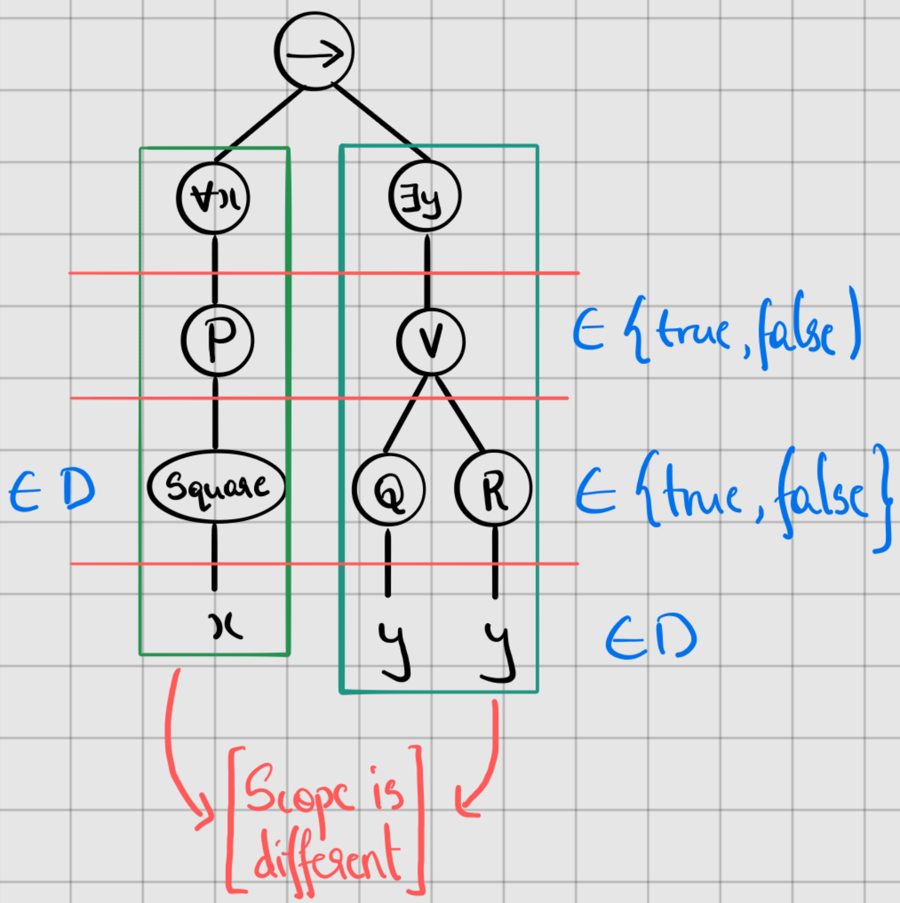
```
    if P:
        break
    if not P:
        break
if P:
    break
return P
```

```
def R(x):
    for y in D:
        if not Q(x,y):
            return false
    return true
```

```
        return true
    return false
```

Parse Trees:-

$$\forall x\, P(x^2) \longrightarrow \exists y\, (Q(y) \vee R(y))$$



Well formed first order formula:

Def-1 :- (Term)

1. Any member of the domain is a term

2. Any variable denoting a member of the domain is a term.
3. If $f: D^k \to D$ is a k-ary function, and $t_1, t_2, \dots, t_k$ are terms, then $f(t_1, t_2, \dots, t_k)$ is a term.

Def-2 :- (Well formed 1st-order formula)

1. Any WFPF is a WFFF.
2. If P is a k-ary predicate, i.e; $P: D^k \to \{true, false\}$ and $t_1, \dots, t_k$ are terms the $P(t_1, \dots, t_k)$ is a WFFF.
3. If $\emptyset$ is a WFFF, $\neg Q$ is a WFFF
4. If $\emptyset, \psi$ are WFFF, then $(\emptyset \wedge \psi), (\phi \vee \psi), (\phi \to \psi), (\phi \leftrightarrow \psi)$ are all WFFF
5. If ϕ is a WFFF, x is a domain variable, $\forall x \phi, \exists x \phi$ are WFFF.
6. Nothing else is a WFFF.

* Presidence Rule :- $\forall x, \exists x, \neg, \wedge, \vee, \to, \leftrightarrow$