1.
(a)
Predicate(s):

- Student(x) - This states that x is a student.
- JEE_Advanced(x) - This represents that x has a JEE Advanced Score.

Function(s):

- None

Variables :

- x

The predicate logic can be written as:

$$\forall x\,(\text{Student}(x) \implies \text{JEE\_Advanced}(x))$$

(b)
Predicate(s):

- Password(x) - This states that x is a password.
- Special(s) - This states that s is a Special Character.
- Contains(x, s) - This states that x contains s.

Function(s):

- None.

Variables :

- x, s

The predicate logic can be written as:

$$\forall x\,(\text{Password}(x) \implies (\exists s(\text{Special}(s) \land \text{Contains}(x, s))))$$

(c)
Predicate(s):

- Award(x) - This states that x is an Award.
- SecondYear(s) - This states that s is a Second Year Student.
- Won(x, s) - This states that s won x.

Function(s):

- None.

Variables :

- x, s

The predicate logic can be written as:

$$\forall x \forall s \, (\text{Won}(x, s) \implies (\text{SecondYear}(s) \land \text{Award}(x)))$$

(d)
Predicate(s):

- Convicted(x) - This states that x is convicted.
- Judge(x) - This states that x is a Judge.

Function(s):

- None.

Variables :

- x

The predicate logic can be written as:

$$\forall x \, (\text{Convicted}(x) \implies \neg \text{Judge}(x))$$

(e)
Predicate(s):

- CSstudent(x) - This states that x is a CS Student.
- LearnsLogic(x) - This states that x learns Mathematical Logic.

Function(s):

- None.

Variables :

- x

The predicate logic can be written as:

$$\forall x \, (\text{CSstudent}(x) \implies \text{LearnsLogic}(x))$$

(f)
Predicate(s):

- Student(x) - This states that x is a Student.
- RollNumber(x, r) - This states that x has a Roll Number r.
- Equals(x, y) - This states that $x = y$.

Function(s):

- None.

Variables :

- x, r1, r2

The predicate logic can be written as:

$$\forall x \forall r1 \forall r2\, (\text{Student}(x) \wedge \text{RollNumber}(x, r1) \wedge \text{RollNumber}(x, r2) \implies Equals(r1, r2))$$

(g)
Predicate(s):

- Bipartite(G) - This states that G is a Bipartite Graph.
- Partition($G, S1, S2$) - This every node of G is either in S1 or S2.
- Disjoint($S1, S2$) - This states that Set $S1$ and Set $S2$ are Disjoint.

Function(s):

- None

Variables :

- G, S1, S2

The predicate logic can be written as:

$$\forall G(Bipartite(G) \implies \exists S1 \exists S2(Partition(G, S1, S2) \wedge Disjoint(S1, S2)))$$

(h)
Predicate(s):

- Clique(C) - This states that set of nodes C is a Clique.
- Edge(x, y) - This states that node x and node y are connected.

Function(s):

- None

Variables :

- C, n1, n2

The predicate logic can be written as:

$$\forall C(Clique(C) \Leftrightarrow \forall n1 \forall n2(((n1 \in C) \wedge (n2 \in C) \wedge (n1 \neq n2)) \implies Edge(n1, n2)))$$

(i)
Predicate(s):

- divByTwo(x) - This states that x is divisible by 2.
- Even(x) - This states that x is Even.

Function(s):

- None

Variables :

- x

The predicate logic can be written as:

$$\forall x(divByTwo(x) \implies Even(x))$$

(j)
Predicate(s):

- Human(x) - This states that x is a Human.
- Spouse(x) - This states that x has a Spouse.

Function(s):

- None

Variables :

- x

The predicate logic can be written as:

$$\exists x(Human(x) \land \neg Spouse(x))$$

2.
(a)Inductive Definition of $Univ(\phi)$ – Universally Quantified Variables
Base Case :

- **Atomic Formula :**
  - If ϕ is a predicate P(t1,t2,...,tn), then:

$$Univ(\phi) = \Phi$$

  - Since atomic formulas have no quantifiers, the set is empty.

Inductive Cases :

- **Negation :**
  - If ϕ is $\neg\psi$:

$$Univ(\phi) = Univ(\psi)$$

- **Binary Operation :**
  - If ϕ is $(\psi 1 \land \psi 2)$ or $(\psi 1 \lor \psi 2)$ or $(\psi 1 \implies \psi 2)$:

$$Univ(\phi) = Univ(\psi 1) \cup Univ(\psi 2)$$

- **Extistential Quantifier :**
  - If ϕ is $\exists x \psi$:

$$Univ(\phi) = Univ(\psi)$$

- **Universal Quantifier :**
  - If ϕ is $\forall x \psi$:

$$Univ(\phi) = Univ(\psi) \cup \{x\}$$

(b) Inductive Definition of $Pred(\phi)$ − Set of Predicates
Base Case :

- **Atomic Formula :**
  - If ϕ is a predicate P(t1,t2,...,tn), then:

$$Pred(\phi) = \{P\}$$

Inductive Cases :

- **Negation :**
  - If ϕ is $\neg \psi$:

$$Pred(\phi) = Pred(\psi)$$

- **Binary Operation :**
  - If ϕ is $(\psi 1 \land \psi 2)$ or $(\psi 1 \lor \psi 2)$ or $(\psi 1 \implies \psi 2)$:

$$Pred(\phi) = Pred(\psi 1) \cup Pred(\psi 2)$$

- **Extistential Quantifier :**
  - If ϕ is $\exists x \psi$:

$$Pred(\phi) = Pred(\psi)$$

- **Universal Quantifier :**
  - If ϕ is $\forall x \psi$:

$$Pred(\phi) = Pred(\psi)$$

3.
(a)PFA the attached photo that has the Syntax Tree

(b)

- **Subformulas :**

– ϕ, Q(x), $\exists z \forall y(((P(f(x), z) \wedge Q(a)) \vee \forall x(R(y, z, g(x))))$,
$(P(f(x), z) \wedge Q(a)), \forall x(R(y, z, g(x))$,
$P(f(x), z), Q(a), R(y, z, g(x))$

- **Terms :**
  - Variables : x, y, z
  - Functions : f^1, g^1
  - Constants : a

- **Predicates :**
  - P(x, y), Q(x), R(x, y, z)

- **Free Variable Occurrences :**
  - None

(c)Interpretation and Environment

Domain (D) = Natural Numbers

Interpretation (I) :

- $f(x) = x, g(x) = x^3 + 1$
- $P(x, y) = (x + y > 0)$
- $Q(x) = (x\%2 = 0)$
- $R(x, y, z) = (x + y + z > 0)$
- $a = 5$
- '+' is the addition operator

Environment (E) : There are no free variables in this language, so there is no need to explicitly define an environment.

4.
(a)

- In N
  x + y = x holds in N only if y = 0.
  If y $\neq$ 0, then x+y $\neq$ x, so the sentence holds.
  Hence True.

- In Z
  x + y = x holds in Z only if y = 0.
  If y $\neq$ 0, then x+y $\neq$ x, so the sentence holds.
  Hence True.

- In R
  x + y = x holds in R only if y = 0.
  If y $\neq$ 0, then x+y $\neq$ x, so the sentence holds.
  Hence True.

(b)

- In N
  x * y = x holds in N if
  x = 0 or y = 1
  In the case x = 0, y = 2 is possible, Hence False.

- In Z
  x * y = x holds in Z if
  x = 0 or y = 1
  In the case x = 0, y = 2 is possible, Hence False.

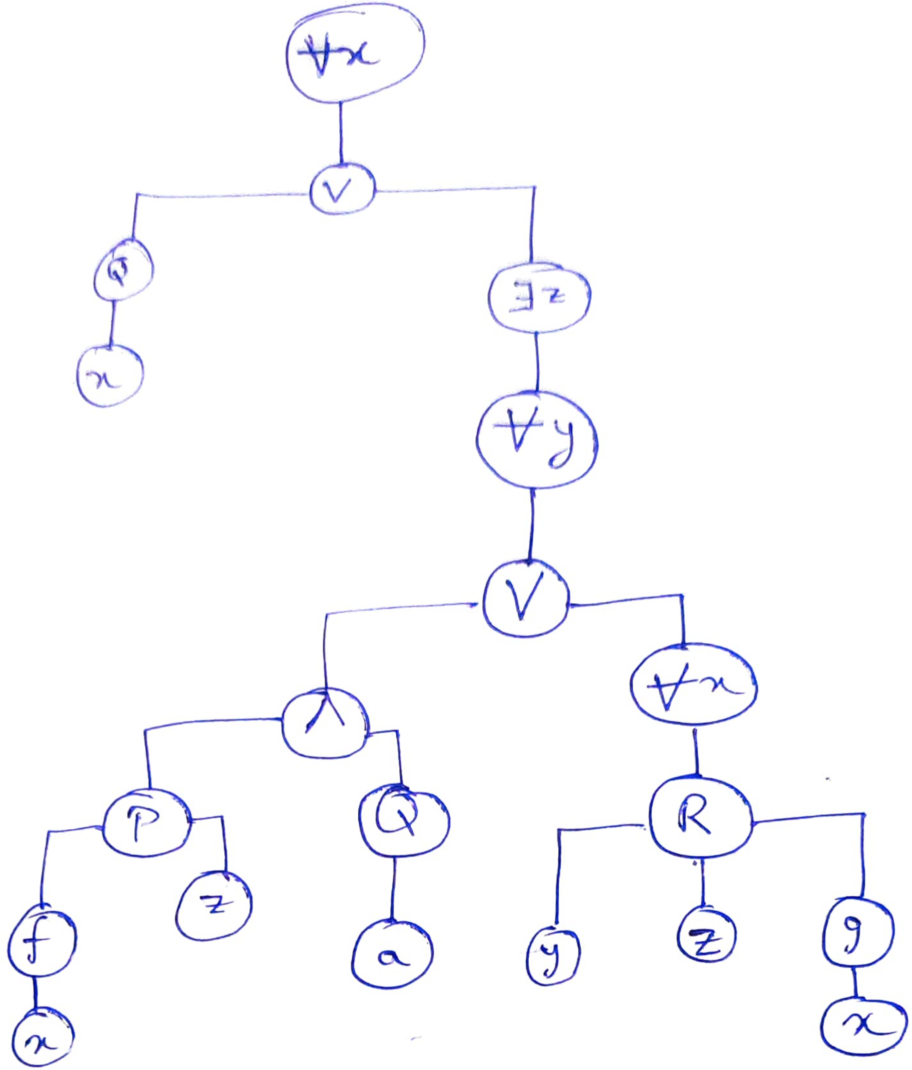
- In R
  x * y = x holds in R if
  x = 0 or y = 1
  In the case x = 0, y = 2 is possible, Hence False.

(c)

- In N
  Let us set x = 2.
  The statement x*y = x+y, holds for y = 2, satisfying y = x. For y $\neq$ 2, the statement x*y = x+y doesn't hold.
  Hence True.

- In Z
  Let us set x = 2.
  The statement x*y = x+y, holds for y = 2, satisfying y = x. For y $\neq$ 2, the statement x*y = x+y doesn't hold.
  Hence True.

- In R
  Let us set x = 2.
  The statement x*y = x+y, holds for y = 2, satisfying y = x. For y $\neq$ 2, the statement x*y = x+y doesn't hold.
  Hence True.



∀x
∨
Q
x
∃z
∀y
∨
∧
∀x
P
Q
R
f
z
a
y
z
g
x
x