Define multisets. How do they differ from ordinary sets? Explain with examples.

Definition of Multisets

A multiset is a collection of elements where repetition is allowed, and each element can appear multiple times. The number of times an element occurs in a multiset is called its multiplicity. Unlike ordinary sets, the identity of the multiset depends not only on which elements are present but also on how many times each element appears.

Difference from Ordinary Sets

Ordinary sets do not allow duplicate elements. Even if an element is listed multiple times, it is counted only once. Sets focus only on the presence or absence of elements, ignoring multiplicity. A multiset, however, treats repeated elements as meaningful, so two multisets can be different even if they contain the same types of elements but in different quantities.

Example to Illustrate the Difference

For a multiset such as {2, 2, 3, 4}, the element 2 appears twice, and this repetition is part of its structure. But in an ordinary set, writing {2, 2, 3, 4} still represents the same set {2, 3, 4}, since duplicates are not counted. Similarly, the multiset {a, a, b} is different from {a, b}, while in normal set notation, both would reduce to {a, b}.

Explain the concept of Boolean algebra. List and explain its fundamental axioms and important theorems.

Concept of Boolean Algebra

Boolean algebra is a branch of mathematics that deals with logical operations and binary variables. Each variable in Boolean algebra takes only two values: 0 or 1, representing false and true. It is mainly used in digital circuits, computer logic, switching theory, and decision-making systems. Boolean algebra provides rules and laws to simplify logical expressions, making the design of digital circuits easier and more efficient.

Fundamental Axioms of Boolean Algebra

Boolean algebra is based on Huntington's axioms, which define how logical operations behave. According to these axioms, every Boolean algebra has two binary operations (AND and OR), one unary operation (NOT), and two special constants (0 and 1). The axioms state that the operations are closed, commutative, associative, and follow identity and distributive laws. They also include the complement law, which ensures that every element has a unique complement such that combining an element with its complement using OR gives 1, and with AND gives 0.

Important Theorems of Boolean Algebra

Boolean algebra contains several theorems that help in simplifying expressions. The idempotent laws explain that repeating an element doesn't change its value, such as A + A = A and A · A = A. The absorption laws show how one term can absorb another, like A + (A · B) = A. The De Morgan's theorems provide a method to transform AND into OR and vice versa, stating that (A · B)′ = A′ + B′ and (A + B)′ = A′ · B′. Other useful theorems include the involution law, which states that double negation returns the original value, and the dominance laws, which show that OR with 1 always results in 1, while AND with 0 always results in 0. Together, these axioms and theorems form the foundation for analyzing and simplifying logical expressions in digital systems.

What are the different operations that can be performed on functions? Explain with examples.

Different Operations on Functions

Several operations can be performed on functions, allowing us to create new functions from existing ones. These include addition, subtraction, multiplication, division, and composition. Each operation combines two functions in a specific way and produces a new function with its own behavior.

Addition and Subtraction of Functions

If f(x) and g(x) are two functions, their sum is defined as (f + g)(x) = f(x) + g(x), and their difference is defined as (f – g)(x) = f(x) – g(x). For example, if f(x) = x^2 and g(x) = 3x, then (f + g)(x) = x^2 + 3x, and (f – g)(x) = x^2 – 3x.

Multiplication and Division of Functions

The product of two functions is defined as (f · g)(x) = f(x)g(x). For example, if f(x) = x and g(x) = x + 1, then (f · g)(x) = x(x + 1) = x^2 + x. The division of functions is defined as (f / g)(x) = f(x) / g(x), but it is valid only when g(x) ≠ 0. For instance, if f(x) = x^2 and g(x) = x, then (f / g)(x) = x^2 / x = x, for x ≠ 0.

Composition of Functions

Composition combines two functions by applying one function inside another. It is defined as (f ∘ g)(x) = f(g(x)). For example, if f(x) = x^2 and g(x) = x + 2, then (f ∘ g)(x) = f(x + 2) = $(x + 2)^2$. This operation is important because it allows us to build more complex functions from simpler ones. These operations help in transforming, analyzing, and understanding functions in various areas of mathematics and computer science.

Write short notes on the following:
(i) Karnaugh Map (K-Map)
(ii) Basic Logic Gates (AND, OR, NOT, NAND, NOR, XOR, XNOR)

i) Karnaugh Map (K-Map)

A Karnaugh Map is a graphical tool used to simplify Boolean expressions by organizing truth table values into a grid where adjacent cells differ by only one bit. This arrangement helps in identifying groups of 1s that can be combined to eliminate variables and produce a simpler Boolean expression. K-Maps are commonly used for 2, 3, or 4 variables and greatly reduce the complexity of logical circuits by minimizing the number of gates required.

(ii) Basic Logic Gates

AND Gate

The AND gate gives an output of 1 only when all of its inputs are 1. If any input is 0, the output becomes 0. It represents logical multiplication. For example, A AND B is 1 only when A = 1 and B = 1.

OR Gate

The OR gate outputs 1 when at least one of its inputs is 1. If all inputs are 0, only then the output is 0. It represents logical addition. For example, A OR B is 1 if either A = 1 or B = 1.

NOT Gate

The NOT gate has a single input and produces the opposite value of that input. If the input is 1, it outputs 0; if the input is 0, it outputs 1. It is also called an inverter.

NAND Gate

The NAND gate is the complement of the AND gate. It outputs 0 only when all inputs are 1; otherwise, the output is 1. It is widely used because all logic circuits can be built using only NAND gates.

NOR Gate

The NOR gate is the complement of the OR gate. It outputs 1 only when all inputs are 0. If any input is 1, the output becomes 0. Like NAND, NOR is also a universal gate.

XOR Gate

The XOR (Exclusive OR) gate outputs 1 when the inputs are different from each other. If both inputs are the same, the output becomes 0. For example, A XOR B is 1 only when one of them is 1 and the other is 0.

XNOR Gate

The XNOR (Exclusive NOR) gate gives 1 when both inputs are the same and 0 when the inputs differ. It is the complement of the XOR gate and is often used in equality checking circuits.

What are Venn diagrams? How are they used to represent set operations?

Venn Diagrams

A Venn diagram is a picture made using circles to show the relationship between sets. Each circle represents a set, and overlapping areas show common elements. They help us easily understand set operations.

Use in Set Operations

Union (A ∪ B) – All elements in A or B.

```
  _______
 /  A      \____
/       B         \
\                _/
 \__________/
```

Intersection (A ∩ B) – Elements common to both A and B.

```
  ___ ___
 /  X     \
/ A∩B   \
\  X      /
 \______/
```

Difference (A – B) – Elements in A but not in B.

```
  _______
/ A |     \
|     |B    |
\____|__/
```

Complement (A') – Everything outside set A.

[Universal Set]

```
  _______
 |  A      |
 |_______|
```

(Shaded outside A)

Define a lattice. Explain the difference between bounded and distributive lattices.

A lattice is an algebraic structure where every pair of elements has a unique least upper bound (called join, denoted by ∨) and a unique greatest lower bound (called meet, denoted by ∧). In simple words, a lattice is a partially ordered set in which any two elements can always be combined to find their common maximum below them and minimum above them.

Difference Between Bounded and Distributive Lattices

A **bounded lattice** is a lattice that has two special elements: a greatest element called *1 (top)* and a least element called *0 (bottom)*. These elements act as limits for the lattice. For example, in a bounded lattice, for every element x, we have 0 ≤ x ≤ 1. Without these two boundary elements, a lattice is not considered bounded.

A **distributive lattice** is a lattice in which the meet and join operations follow distributive laws. This means the join distributes over the meet and the meet distributes over the join. Formally, a lattice is distributive if for all elements a, b, and c, the relations
a ∧ (b ∨ c) = (a ∧ b) ∨ (a ∧ c)
and
a ∨ (b ∧ c) = (a ∨ b) ∧ (a ∨ c)
hold true

What are cosets in group theory? Illustrate with an example.

Cosets are subsets formed by adding or multiplying every element of a subgroup with a fixed element of the main group. They help in understanding the structure of groups and dividing a group into equal-sized parts. If *H* is a subgroup of a group *G* and *a* is any element of *G*, then the set formed by combining *a* with each element of *H* is called a coset.

A **left coset** of *H* in *G* is written as **aH = {a·h | h ∈ H}**, and a **right coset** is written as **Ha = {h·a | h ∈ H}**.

Example

Consider the group **G = (ℤ, +)**, the set of all integers under addition, and the subgroup **H = {…, –6, –3, 0, 3, 6, …}**, which is the set of multiples of 3.

Take **a = 1**.

The **left coset** of H with element 1 is:
1 + H = {1 + h | h ∈ H}
= {…, –5, –2, 1, 4, 7, …}
This means every element of H is shifted by 1.

Similarly, take **a = 2**.
2 + H = {2 + h | h ∈ H}
= {…, –4, –1, 2, 5, 8, …}

So the integers are divided into cosets based on remainders when divided by 3:

- Numbers giving remainder 0 → H
- Numbers giving remainder 1 → 1 + H
- Numbers giving remainder 2 → 2 + H

Define graph theory. What are the types of graphs?

Graph Theory

Graph theory is a branch of mathematics that studies relationships between objects. A graph consists of **vertices (nodes)** and **edges (lines)** connecting pairs of vertices. Graphs are used to model real-life problems such as networks, routes, social connections, communication systems, and computer structures.

Types of Graphs

A **simple graph** has no loops and no multiple edges between the same vertices. A **multigraph** allows multiple edges between the same pair of vertices. A **pseudograph** includes both loops and multiple edges. In a **directed graph (digraph)**, edges have a direction, while in an **undirected graph**, edges do not have direction. A **weighted graph** assigns a numerical weight to each edge, often used for distance or cost. A **complete graph** connects every pair of vertices by an edge. A **connected graph** has a path between every pair of vertices, whereas a **disconnected graph** does not. Finally, a **bipartite graph** divides its vertices into two sets such that no two vertices within the same set are connected.

What is a tree in graph theory? Explain its properties.

A **tree** in graph theory is a connected graph with no cycles. It is the simplest structure that connects all its vertices without forming any loops.

Properties of a Tree

A tree with **n vertices** always has **n – 1 edges**. There is exactly **one unique path** between any two vertices. Adding any new edge creates a cycle, and removing any existing edge makes the graph disconnected. Every tree has at least **two leaves**, which are vertices of degree one. Trees are widely used in data structures, hierarchy representation, and network design.

Define permutation and combination with suitable examples.

A permutation is an arrangement of objects **where order matters**. Changing the order creates a different permutation.

For example, using the letters A, B, C:
ABC, ACB, BAC, BCA, CAB, CBA are different permutations because the order changes each time.

Combination

A combination is a selection of objects **where order does not matter**. Changing the order does not create a new combination.

For example, choosing 2 letters from A, B, C gives:
AB, AC, BC.
Here, AB and BA are considered the same combination.

In simple terms, **permutation = arrangement with order**, and **combination = selection without order**.

What is a truth table? Construct a truth table for (p ∧ q) ∨ ¬r.

A truth table is a table that shows all possible truth values of logical variables and the resulting value of a logical expression. It helps in understanding how a logical statement behaves under every possible case.

Truth Table for (p ∧ q) ∨ ¬r

Here p, q, r can be either True (T) or False (F).

p	q	r	p ∧ q	¬r	(p ∧ q) ∨ ¬r
T	T	T	T	F	T
T	T	F	T	T	T
T	F	T	F	F	F
T	F	F	F	T	T
F	T	T	F	F	F
F	T	F	F	T	T
F	F	T	F	F	F
F	F	F	F	T	T

What is a partially ordered set (poset)? Give an example.

A **partially ordered set (poset)** is a set combined with a relation (usually denoted ≤) that is **reflexive**, **antisymmetric**, and **transitive**. This relation tells us how elements are ordered, but not all elements must be comparable. That means some elements may have no ordering between them.

Example

Consider the set **A = {1, 2, 3, 6}** with the relation **"divides" (|)**.
Here, 1 | 2, 1 | 3, 1 | 6, 2 | 6, and 3 | 6.
This forms a **poset** because the divides relation is reflexive, antisymmetric, and transitive, but not all elements compare directly (for example, 2 and 3 do not divide each other).