Computer science

Chapter 1

Q 1. What is Data?

Data are raw facts or numbers.
They have no meaning by themselves.

Example: 10, 25, 50

Q 2. What is Information?

Information is useful data.
It has meaning.

Example: “The total marks are 85.”

Q 3. What is Data Processing / Computing?

Data processing means changing raw data into useful information.

Example: Adding marks to find total.

Q 4. What is Data Capturing?

Data capturing means collecting data.

Example: Writing students’ marks in a register.

Q 5. What is Data Manipulation?

Data manipulation means changing or editing data.

Example: Correcting wrong marks.

Q 6. What is Sorting?

Sorting means arranging data in order.

Example: Arranging numbers from small to big.

Q 7. What is Classification?

Classification means grouping similar things together.

Example: Grouping students by grade (A, B, C).

Q 8. What is Summarizing?

Summarizing means giving short and important information from large data.

Example: Finding total or average marks.

Q 9. What is Calculating?

Calculating means doing math on data.

Example: Adding, subtracting, multiplying.

Q 10. What is Storage?

Storage means saving data for later use.

Example: Saving a file on a computer.

Q 11. What is Retrieval?

Retrieval means getting back saved data.

Example: Opening a saved file.

Q 12. What is Reproduction?

Reproduction means making copies.

Example: Printing a document.

Q 13. What is Communication?

Communication means sharing information with others.

Example: Sending a message or email.

Q 14. What is field?

A **field** is one small piece of information.

Example:
In a student record:
Name = Ali
Age = 15
Each item (Name, Age) is a field.

Q 15. What is Record?

A **record** is a group of related fields about one person or thing.

Example:
Name: Ali
Age: 15
Class: 9

All together make one record.

Q 16. What is File?

A **file** is a collection of many records.

Example:
All students' records in one list = Student file.

Q 17. What is Master File?

A **master file** stores main and permanent data.

Example:
A file that keeps all student details for the whole year.

Q 18. What is Transaction File?

A **transaction file** stores daily changes or updates.

Example:
Daily attendance or new fee payment records.

Q 19. What is Backup File?

A **backup file** is a copy of data kept for safety.

Example:
Saving a copy of student data in case the computer crashes.

Q 20. What is Data File?

A **data file** stores data or information.

Example:
A file that contains students' marks.

Q 21. What is Program File?

A **program file** contains instructions for the computer.

Example:
A file that tells the computer how to calculate marks.

Q 22. What is Sequential File?

A **sequential file** stores data in order, one after another.

Example:
Reading names from top to bottom in a list.

Q 23. What is Random Access File?

A **random-access file** allows you to find any record directly.

Example:
Searching a student by roll number quickly.

Q 24. What is Indexed Sequential File?

An **indexed sequential file** stores data in order and also uses an index to find data fast.

Example:
A phone book (names in order + index to search quickly).

Q 25. What is Database?

A **database** is a collection of related data stored in a computer.

Example:
A school database stores students' names, marks, and fees.

Q 26. What is Database Objectives?

Database objectives are the goals of using a database.

Main goals:

- Keep data safe
- Reduce duplicate data
- Make data easy to find
- Keep data correct

Q 27.What is Database Models?

Database models show how data is organized in a database.

Simple types:

- Hierarchical (like a tree)
- Network (like a web)
- Relational (tables with rows and columns)

Example:
In relational model, data is stored in tables.

Q 28. What is DBMS (Database Management System)?

A **DBMS** is software that manages a database.

It helps to:

- Store data
- Edit data
- Delete data
- Find data

Example: MySQL, Oracle

Q 29.What is DBMS Objectives?

DBMS objectives are the goals of using DBMS.

Main goals:

- Store large data easily
- Reduce data repetition
- Keep data secure
- Allow many users to use data

Q 30.What is Data Dictionary?

A **data dictionary** is a file that stores information about data.

It tells:

- Field names
- Data types

- Size of fields

It is like a guide book for the database.

Q 31. What is Query Language?

A **query language** is used to ask questions from a database.

Example:
SQL is used to find student marks.

Q 32.What is Report Generator?

A **report generator** is a tool that creates reports from data.

Example:
Printing a student result sheet.

Q 33. What is Data Redundancy?

Data redundancy means repeating the same data again and again.

Example:
Saving the same student name many times in different places.

Q 34. What is Data Inconsistency?

Data inconsistency means data does not match.

Example:
A student's phone number is different in two files.

Chapter 2

Q 1.What is Entity?

An **entity** is a person, place, or thing about which we store data.

Example:
Student, Teacher, Book

Q 2.What is Attributing?

An **attribute** is a quality or detail of an entity.

Example:
For Student entity:
Name, Roll No, Age are attributes.

Q 3. Fixed Length Field?

A **fixed length field** stores data of the same size every time.

Example:
Roll number always 5 digits (12345).

Even if number is small, space is fixed.

Q 4.What is Variable Length Field?

A **variable length field** stores data of different sizes.

Example:
Name field (Ali, Muhammad Ali, etc.)

It uses only the space needed.

Q 5. What is View?

A **view** is a part of a database shown to a user.

Example:
A teacher can see marks but not student fees.

Q 6. What is Index?

An **index** helps to find data quickly.

Example:
Like index page of a book to find topics fast.

Q 7. What is User?

A **user** is a person who uses the database.

Example:
Student, teacher, clerk.

Q 8. What is DA (Data Administrator)?

DA is the person who plans and controls the data.

He decides:

- What data to store
- How data should be organized

Q 9. What is DBA (Database Administrator)?

DBA is the person who manages and controls the database system.

He:

- Keeps data safe
- Gives user access
- Solves database problems

Q 10. What is Relation / Table?

A **relation** is also called a **table** in a database.

It stores data in the form of:

- Rows
- Columns

Example:

Roll No	Name	Class
101	Ali	9

This whole table is called a **relation**.

Q 11. What is Properties of Relation?

A relation (table) has these simple rules:

1. Each table has a unique name.
2. Each column has a different name.
3. Each row is unique (no duplicate rows).
4. Data is stored in rows and columns.
5. Order of rows does not matter.

Q12. What is Key?

A **key** is a field (or column) that is used to identify a record uniquely.

Example:
Roll Number can be a key because every student has a different roll number.

13. Types of Key

1. Primary Key

The main key used to identify each record.

Example: Roll No

- Cannot be empty
- Cannot be repeated

2. Candidate Key

A field that can become a primary key.

Example:
Roll No or ID Card Number

Both can identify a student.

3. Alternate Key

A candidate key that is not selected as primary key.

If Roll No is primary key,
ID Card No becomes alternate key.

4. Foreign Key

A key that connects two tables.

Example:
Student table and Fee table connected by Roll No.

Chapter 3

Q 1. What is Feasibility Study?

It means checking **"Can we do this project or not?"**

We check:

- Do we have enough money?
- Do we have the right technology?
- Do we have enough time?

If yes → Project can start.

Q 2. What is Requirements Analysis?

It means understanding **what the customer wants**.

Example:
If making a school system, we ask:

- Do you need student details?
- Do you need marks and attendance?

Q 3.What is Project Planning?

It means making a **plan** before starting work.

We decide:

- What work to do
- Who will do it
- When to finish it

Q 4. What is Data Analysis?

It means studying data carefully.

Example:
Looking at students' marks to find:

- Highest marks
- Average marks

Q 5.What is Data Modeling?

It means drawing a **design of data** before making a database.

It shows:

- What data we will store
- How data is connected

Q 6. What is Relationship?

It means a **connection between two tables**.

Example:
Student and Course are connected because students join courses.

Q 7. Types of Relationship

1 One-to-One (1:1)

Meaning:

One thing is connected to only one thing.

Very Simple Example:

- One person → One passport
- One student → One roll number

This means:

- One person cannot have two passports.
- One passport belongs to only one person.

So, one matches with only one.

2 One-to-Many (1:M)

Meaning:

One thing is connected to many things.

Very Simple Example:

- One teacher → Many students
- One mother → Many children
- One customer → Many orders

This means:

- One teacher teaches many students.
- But each student has only one class teacher.

So, one side has one, the other side has many.

This is the **most common type** in database.

3 Many-to-Many (M: N)

Meaning:

Many things are connected to many things.

Very Simple Example:

- Many students → Many subjects

This means:

- One student studies many subjects.
- One subject is studied by many students.

Both sides have many.

In database, we usually create a **third table** to connect them.

Example:
Student table
Subject table
Enrollment table (connects both)

Q 7. What is Logical Design?

- **What it is:** This is the plan of how data will be stored in the database **without worrying about how it will actually be implemented physically**.
- **Example:** You decide that a school database will have tables for **Students, Teachers, and Classes**, and how these tables relate to each other.

Q 8. What is Physical Design?

- **What it is:** This is about **how data is actually stored in the computer**. It focuses on storage details, indexes, and performance.
- **Example:** Deciding that the Student table will be stored in a **B-Tree index** for faster searching.

Q 9.What is Data Distribution Strategies?

- **What it is:** How data is stored across **different computers or servers**.
- **Types:**

1. **Centralized:** All data in one place.
2. **Replicated:** Copies of data stored on multiple servers.
3. **Fragmented:** Data split into pieces and stored in different places.

Example: A bank might store customer data in different city branches but also keep a central copy.

Q 10. What is File Organization?

- **What it is:** The way **data files are arranged on storage** so that retrieval is efficient.
- **Common Methods:**
  1. **Heap File:** Unordered, simple storage.
  2. **Sequential File:** Data stored in order (e.g., alphabetically).
  3. **Hashed File:** Data stored using a hash function for quick access.

Q 11.What is Integrity Constraints?

- **What it is:** Rules to **keep the data correct and reliable**.
- **Types:**
  1. **Primary Key:** Unique ID for each record (like student ID).
  2. **Foreign Key:** Link between tables (like student ID in the marks table).
  3. **Not Null:** Field must have a value (like a student must have a name).
  4. **Check:** Certain conditions must be met (age > 5).

Q12. What is Implementation?

- **What it is:** The process of **actually creating the database and running it** using a Database Management System (DBMS).
- **Example:** Writing SQL commands to create tables, set rules, and insert data.

Chapter 4

Q 1.What is Data Integrity?

Making sure data in the database is **correct, consistent, and reliable**.

- **Example:** A student's age cannot be negative.

Q 2. What is Normalization?

A process of **organizing data in tables** to reduce duplication and errors.

- **Example:** Instead of storing the teacher's name in every student record, store it in a separate Teacher table.

Q 3. What is Normal Form?

Rules to check how clean and organized a table is.

- **Types (basic ones):**
    1. **1NF (First Normal Form):** No repeating groups; each column has a single value.
    2. **2NF (Second Normal Form):** No partial dependency on part of a key.
    3. **3NF (Third Normal Form):** No dependency on non-key columns.

Q 4. What is Homonym?

Same name, different meaning in database terms.

- **Example:** Two tables both have a column called "ID," but one is Student ID and one is Teacher ID.

Q 5. What is Synonym?

Different names, same meaning.

- **Example:** "EmployeeID" in one table and "StaffID" in another mean the same thing.

Q 6. What is Mutual Exclusion?

Two things cannot happen at the same time in the database.

- **Example:** A seat in a bus booking system cannot be booked by two people at the same time.

Q 7. What is Redundancy?

Duplicate data that wastes space and can cause errors.

- **Example:** Storing the same student address in 10 different tables.

Q 8. What is Repeating Group?

A set of multiple values in a single column, which is not allowed in normalized tables.

- **Example:** Storing multiple phone numbers in one column separated by commas.

Q 9. What is Functional Dependency (FD)?

- **Definition:** When one column (or set of columns) in a table **determines another column**.
- **Example:** In a table of students:

StudentID	Name	Age
101	Alice	20

Here, **StudentID → Name** means if you know the StudentID, you can find the Name.

Easy way to remember: **A → B** means **A determines B**.

Q 10.What is Partial Functional Dependency?

- **Definition:** When a column depends **only on part of a composite primary key** (not the whole key).
- **Example:**

Table: (StudentID, CourseID → Grade, StudentName)

- Primary key = (StudentID, CourseID)
- **StudentName depends only on StudentID**, not CourseID → This is **partial dependency**.

Q 11 .What is 1NF (First Normal Form)?

Definition: Table must have **no repeating groups**; each column contains only atomic (single) values.

- **Example (Not 1NF):**

StudentID	Courses
101	Math, Physics

- **Convert to 1NF:**

StudentID	Course
101	Math
101	Physics

Q 12.What is Transitive Functional Dependency?

Definition: When **A → B** and **B → C**, then **A → C** is a **transitive dependency**.

- **Example:**

StudentID	DepartmentID	DepartmentName
101	D01	Science

- StudentID → DepartmentID and DepartmentID → DepartmentName
- So, StudentID → DepartmentName (**transitive**)

Q 13.What is 2NF (Second Normal Form)?

Definition:
1NF + **No partial dependencies** (no column depends on part of a composite key).

- **Example:** Using our earlier student-course table:

StudentID CourseID StudentName Grade

- StudentName depends only on StudentID → partial dependency.
- **2NF solution:** Move StudentName to a separate Student table.

Q 14. What is 3NF (Third Normal Form)?

Definition:
2NF + **No transitive dependencies**.

- **Example:** Using Student-Department example:

| StudentID | DepartmentID | DepartmentName |

- DepartmentName depends on DepartmentID, not StudentID → transitive dependency.
- **3NF solution:** Separate tables:

Student Table: StudentID → DepartmentID
Department Table: DepartmentID → DepartmentName

Q 15.What is Anomalies?

These happen when tables are **not normalized**:

1. **Insertion Anomaly:** Cannot insert data without some other data.
   - E.g., Can't add a new department if no student is assigned yet.
2. **Update Anomaly:** Need to update the same info in multiple places.
   - E.g., Change DepartmentName → must update all students in that department.
3. **Delete Anomaly:** Deleting data removes other useful info.
   - E.g., Delete last student in a department → department info lost.

C language

Chapter 9

Q 1. What is Identifier?

A name given to something in a program.
It can be a name for a variable, function, class, etc.

Example:

int age;

Here, **age** is an identifier.

Q 2. What is Standard Identifier?

Names that are already defined by the programming language or its library.

Example in C:

printf();

scanf();

printf is a standard identifier because it is already provided by C.

Q 3. What is User-defined Identifier?

Names created by the programmer.

Example:

int marks;

marks is a user-defined identifier because you created it.

Q 4. What is Keyword?

Special words that have a fixed meaning in a programming language.
You **cannot use them as names**.

Examples in C:

int, float, if, else, return, while

Example:

int number;

Here, int is a keyword.

Q 5.What is Variable?

A container that stores a value.
The value can change.

Example:

int age = 20;

age is a variable.

Q 6. What is Constant?

A value that **does not change**.

Example:

const int x = 10;

Here, 10 is a constant and cannot be changed.

Q 7. What is Datatype?

It tells the computer what type of data is stored.

Examples:

- int → whole numbers (10, 25)
- float → decimal numbers (10.5)
- char → single character ('A')

Example:

float price = 99.5;

Q 8. What is Operator?

A symbol used to perform operations.

Examples:

- + (addition)
- - (subtraction)
- * (multiply)
- / (divide)

Example:

int sum = 5 + 3;

Q 9. What is Operator Precedence?

The order in which operations are done.

Example:

2 + 3 * 4

Multiplication happens first →
Answer = 2 + 12 = 14

So * has higher precedence than +.

Q 10. What is Operator Associativity?

When two operators have the same precedence, associativity decides the direction (left to right or right to left).

Example:

10 - 5 – 2

Subtraction works left to right:

(10 - 5) - 2 = 3

Q 11.What is Expression?

Combination of variables, values, and operators that gives a result.

Example:

a + b * 2

This is an expression.

Q 12.What is Comment?

Notes in the program.
Computer ignores comments.

Example in C:

// This is a comment

Or

/* This is

a multi-line comment */

Q 13. What are the Rules for Naming a Variable?

A variable is a **name used to store data**.

1. Start with a letter or _

✔ `age`
✔ `_total`
✘ `1age` (cannot start with number)

2. No spaces

✔ `totalMarks`
✘ `total marks`

3. Do not use keywords

✘ `int`
✘ `float`
These are special words in programming.

4. Only letters, numbers, and _

✔ `marks1`
✔ `student_name`
✘ `marks@`

5. Names are case-sensitive

`age` and `Age` are different.

Q 14. Types of Operators

Operators are symbols used to perform operations.

1. Arithmetic Operators

Used for calculations.

Operator	Meaning
+	Addition

Operator	Meaning
-	Subtraction
*	Multiply
/	Divide
%	Modulus (remainder)

Example:

5 + 3 = 8

2. Relational Operators

Used to compare values.

Operator	Meaning
==	Equal
!=	Not equal
>	Greater than
<	Less than
>=	Greater or equal
<=	Less or equal

3.Assignment Operators

Used to assign values.

Operator	Meaning
=	Assign 5 to x
+=	x = x + 3
-=	x = x – 2

4.Logical Operators

Logical operators are used to **join two or more conditions**.

They give answer in **True** or **False**.

There are **3 logical operators**:

- **AND Operator (&&)**

Meaning: **Both conditions must be true**

If both are true → Answer is **True**
If one is false → Answer is **False**

Example:

(5 > 3)&& (10 > 7)=True

5 > 3 → True

10 > 7 → True

So, answer = **True**

- **OR Operator (||)**

Meaning: **At least one condition must be true**

If one is true → Answer is **True**
If both are false → Answer is **False**

Example:

(5 > 3) || (10 < 7) =True

First is True
Second is False

So, answer = **True**

- **NOT Operator (!)**

Meaning: **Opposite result**

- True becomes False
- False becomes True

Example:

(5 > 3)! =False

5 > 3 → True
After using NOT → False

Chapter 10

Q 1. What is Signature of a Function?

Function signature means:

- Function name
- Return type
- Parameters (inputs)

Example:

int add (int a, int b)

 int → return type

 add → function name

 int a, int b → parameters

Q 2. What is printf ()?

printf() is used to **print output on screen**.

Example:

printf("Hello");

It shows → Hello

Q 3.What is scanf()?

scanf() is used to **take input from user**.

Example:

scanf ("%d", &x);

It takes number input from user.

Q 4. What is Format String?

Format string tells **what type of data to print or read**.

Example inside printf or scanf:

"%d"

"%f"

"%c"

"%s"

Q 5. What is Format Specifier?

Format specifier shows **data type**.

Common format specifiers:

Specifier	Meaning
%d	Integer
%f	Float
%c	Character
%s	String

Example:

printf ("%d", 10);

Q 6. What is Field-Width Specifier?

Field width means **how much space to print the value**.

Example:

printf ("%5d", 10);

If width is 5, number will take 5 spaces.

Q 7. What is Escape Sequence?

Escape sequences are **special characters** used inside quotes.

They start with \

Escape	Meaning
\n	New line
\t	Tab space
\	Backslash
"	Double quote

Example:

printf("Hello\nWorld");

Output:
Hello
World

Q 8.What is getch()?

getch () takes **one-character input.**
It does NOT show the character on screen.

Q 9.What is getche()?

getche () takes **one-character input.**
It shows the character on screen.

Chapter 11

Q 1. What is Control Structure?

Control structure means **how a program runs step by step**.

It controls the **flow of program**.

There are 3 types:

- Sequence
- Selection
- Repetition

Q 2. What is Sequence Structure?

Instructions run **one by one in order**.

Example:

```
int a = 5;
int b = 3;
int sum = a + b;
```

Steps run from top to bottom.

Q 3. What is Selection Structure?

Used to **make decisions**.

Program chooses one option.

Example:

- if
- if-else
- switch

Q 4□.What is Repetition Structure?

Used to **repeat a task many times**.

Example:

- for loop
- while loop
- do-while loop

Q 5. What is if Statement?

Used to check a condition.

If condition is true → code runs.

Example:

```
if (a > 5)
{
  printf("Big");
}
```

Q 6□.What is if-else Statement?

Used when there are two choices.

If true → first block runs
If false → second block runs

Example:

```
if (a > 5)
{
  printf("Big");
}
else
{
  printf("Small");
}
```

Q 7□.What is Nested if?

An **if inside another if**.

Example:

```
if (a > 0)
{
  if (a > 10)
  {
    printf("Large");
  }
}
```

Q 8□. What is switch Statement?

Used when there are **many choices**.

Example:

```
switch(day)
{
  case 1: printf("Monday"); break;
  case 2: printf("Tuesday"); break;
  default: printf("Invalid");
}
```

Q 9. What is Condition?

A condition is a **test** that gives True or False.

Example:

a > 5

Q 10. What is Compound Condition?

When we use **two or more conditions together**.

We use:

- && (and)
- || (or)

Example:

if (a > 5 && b < 10)

Q 11. What is Conditional Operator (? :)?

Short form of if-else.

Syntax:

condition ? value_if_true : value_if_false;

Example:

int max = (a > b)? a: b;

7. Attempt the following parts:

(i)

Assuming **x is 10.0** and **y is 15.0**, what are the values of the following conditions?

a) x! = y
b) x < x
c) x >= y - x
d) x == y + x - y

Hint: The value of a true condition is 1 and a false condition is 0.

a) **x != y**

10 ≠ 15 → True

Answer: 1

b) **x < x**

10 < 10 → False

Answer: 0

c) **x >= y - x**

y - x = 15 - 10 = 5
10 ≥ 5 → True

Answer: 1

d) **x == y + x - y**

y + x - y = 15 + 10 - 15 = 10
10 == 10 → True

Answer: 1

(ii)

Write an expression to test each of the following relationships:

a) **age** is from 18 to 25.
b) **temperature** is less than 40.0 and greater than 25.0.
c) **year** is divisible by 4.
d) **speed** is not greater than 80.
e) **y** is greater than **x** and less than **z**.
f) **w** is either equal to 6 or not greater than 3.

Note: The bold face words represent variables' names.

a) age is from 18 to 25

Answer:

age >= 18 && age <= 25

b) temperature is less than 40.0 and greater than 25.0

Answer:

temperature < 40.0 && temperature > 25.0

c) year is divisible by 4

Answer:

year % 4 == 0

d) speed is not greater than 80

Answer:

speed <= 80

e) y is greater than x and less than z

Answer:

y > x && y < z

f) w is either equal to 6 or not greater than 3

Answer:

w == 6 || w <= 3

(iii)

Write assignment statement for the following:

a) Assigns a value of 1 (one) to the variable **test** if **k** is in the range –m through +m, inclusive. Otherwise, assigns a value of 0.

b) Assigns a value of 1 (1) to the variable **lowercase** if **ch** is a lowercase letter; otherwise, assigns a value of 0 (0).

c) Assigns a value of 1 (1) to the variable **divisor** if **m** is a divisor of **n**; otherwise, assigns a value of 0 (0).

Note: The bold face words represent variables' names.

a) Assign 1 to test if k is between -m and +m, otherwise 0

Answer:

test = (k >= -m && k <= m) ? 1 : 0;

b) Assign 1 to lowercase if ch is a lowercase letter, otherwise 0

Answer:

```
lowercase = (ch >= 'a' && ch <= 'z') ? 1 : 0;
```

c) Assign 1 to divisor if m is divisor of n, otherwise 0

Answer:

```
divisor = (n % m == 0) ? 1: 0;
```

Example 2:

Write a program that accepts three numbers from the user and displays the largest number.

```
#include <stdio.h>
void main()
{
   int a, b, c;
 printf("Enter three numbers: ");
   scanf("%d %d %d", &a, &b, &c);
 if (a > b)
   {
      if (a > c)
         printf("%d is largest", a);
      else
         printf("%d is largest", c);
   }
   else
   {
```

```
        if (b > c)
            printf("%d is largest", b);
        else
            printf("%d is largest", c);
    }
}
```

Example 3:

Write a program that inputs a number and checks whether it is **positive, negative, or zero**.

```
#include <stdio.h>
void main()
{
    int num;
   printf("Enter a number: ");
    scanf("%d", &num);
if (num > 0)
        printf("The number is positive");
else if (num < 0)
        printf("The number is negative");
   else
        printf("The number is zero");
}
```

Question no 8:

- **Area of a square and triangle**

```
#include <stdio.h>

int main() {
```

```
    char ch;
    float side, base, height, area;

    printf("Enter S for Square or T for Triangle: ");
    scanf(" %c", &ch);

    if (ch == 'S' || ch == 's') {
       printf("Enter side of square: ");
       scanf("%f", &side);
       area = side * side;
       printf("Area of Square = %.2f", area);
    }
    else if (ch == 'T' || ch == 't') {
       printf("Enter base and height of triangle: ");
       scanf("%f %f", &base, &height);
       area = 0.5 * base * height;
       printf("Area of Triangle = %.2f", area);
    }
    else {
       printf("Invalid choice!");
    }

    return 0;
}
```

Question no 9:

- **Leap year or not**

```
#include <stdio.h>

int main () {

   int year;

 printf ("Enter a year: ");

   scanf ("%d", &year);

if ((year % 4 == 0 && year % 100!= 0) || (year % 400 == 0))

      printf ("Leap year");

   else
```

```
  printf ("Not a leap year");
return 0;
}
```

Question no 10:

- **Find Temperature**

```
#include <stdio.h>

int main() {
   float temp;
printf("Enter temperature: ");
   scanf("%f", &temp);
 if (temp > 35)
      printf("Hot day");
   else if (temp >= 25 && temp <= 35)
      printf("Pleasant day");
   else
      printf("Cool day");
     return 0;
}
```

Question no 11:

- **Calculate percentage and display grade**

```
#include <stdio.h>

int main() {
   float marks, percentage;

   printf("Enter obtained marks (out of 1100): ");
   scanf("%f", &marks);
```

```
    percentage = (marks / 1100) * 100;

    printf("Percentage = %.2f%%\n", percentage);

    if (percentage >= 80)
       printf("Grade: A+");
    else if (percentage >= 70)
       printf("Grade: A");
    else if (percentage >= 60)
       printf("Grade: B");
    else if (percentage >= 50)
       printf("Grade: C");
    else if (percentage >= 40)
       printf("Grade: D");
    else if (percentage >= 33)
       printf("Grade: E");
    else
       printf("Grade: F");

    return 0;
}
```

Question no 12:

- **Switch statement**

```
#include <stdio.h>

int main () {
   float num1, num2, result;
   int choice;

   printf("Enter two numbers: ");
   scanf("%f %f", &num1, &num2);

   printf("1. Addition\n2. Subtraction\n3. Multiplication\n4. Division\n");
   printf("Enter your choice: ");
   scanf("%d", &choice);

   switch (choice) {
      case 1:
         result = num1 + num2;
         printf ("Sum = %.2f", result);
         break;

      case 2:
         result = num1 - num2;
         printf ("Difference = %.2f", result);
```

```
            break;

        case 3:
            result = num1 * num2;
            printf("Product = %.2f", result);
            break;

        case 4:
            if (num1 > num2)
                result = num1 / num2;
            else
                result = num2 / num1;

            printf("Division Result = %.2f", result);
            break;

        default:
            printf("Wrong Choice");
    }

    return 0;
}
```

Chapter 12

Q 1. What is Iteration?

Repeating something again and again.
In programming, it means running the same block of code multiple times.

Example:
If you print "Hello" 5 times, that is iteration.

Q2. What is Counter Loop?

A loop that repeats a specific number of times.

It uses a **counter variable** to keep track of how many times it runs.

Example:

for (int i = 1; i <= 5; i++)

This loop runs 5 times.

Q3. What is Sentinel Loop?

A loop that keeps running until a special value is entered.

It does not stop at a fixed number — it stops when a **sentinel value** is given.

Q 4. What is Sentinel Variable?

The variable that stores the user input in a sentinel loop.

It is checked to see if the loop should stop.

Example:

int number;

Here, number can be a sentinel variable.

Q5. What is Sentinel Value?

A special value that tells the loop to stop.

Example:
If the sentinel value is **-1**, the loop stops when the user enters **-1**.

Q6. What is Nested Loop?

A loop inside another loop.

Example:
A clock:

- The hour changes
- Inside that, minutes change

So, one loop runs inside another loop.

Q7. What is Pretest Loop

A loop that checks the condition **before** running.

If the condition is false, it will not run at all.

Example:
while loop

Q8. What is Posttest Loop?

A loop that checks the condition **after** running.

It will run **at least once**.

Example:
do-while loop

Q9. What is Goto Statement?

A command that jumps to another part of the program.

It moves the program directly to a labeled line.

It is not recommended because it can make programs confusing.

Question no 5 Exercise

Write the output of the following program fragments:

- (a) k = 0;
  while (k <= 5)
  {
  printf ("%3d %3d\n", k, 10 - k);
  k++;
  }
  OUTPUT

0	10
1	9
2	8
3	7
4	6
5	5

- (b) j = 10;

  for (int i = 1; i <= 5; ++i) {

  printf ("%d %d\n", i, j);

  j -= 2;}

  OUTPUT

1	10
2	8
3	6
4	4
5	2

Question no 6

(a)Correct the following code segment according the following condition

```
int count = 0, next_num, sum = 0;
    while (count < 5)
   {   count += 1;
       printf ("Next Number -> ");
       scanf ("%d", &next_num);
       sum += next_num;
   }
  printf ("%d numbers were added.\n", count);
   printf("Their sum is %d.\n", sum);
```

Correction:

I. Removed unnecessary semicolon after while
II. Added braces { }
III. Initialized sum
IV. Corrected sum += next_num

(b) Rewrite the following code segment using a do-while statement

```
sum = 0;

odd = 1;
```

```
do
{
   if (odd < n)
      sum = sum + odd;
   odd = odd + 2;
}
while (odd < n);
printf("Sum of the positive odd numbers less than %d is %d\n", n, sum);
```

Question no 7

Trace the output of following code segments

```
(a) f or (k = 1; k <= n; ++k)
{
   for (j = 0; j < k; ++j)
   {
      printf("*");
   }
   printf("\n");
}
```

Output

```
*
**
***
****
*****
```

(b) for (k = n; k > 0; --k)

```
{
   for (j = m; j > 0; --j)
   {
      printf ("*");
   }
   printf("\n");
}
```

Output

```
***
***
***
***
***
```

Q9. Program to check whether a number is prime or not

```
#include <iostream>
using namespace std;

int main() {
   int num, i;
   bool isPrime = true;

   cout << "Enter a number: ";
   cin >> num;

   if (num <= 1)
      isPrime = false;
   else {
      for (i = 2; i <= num / 2; i++) {
```

```
            if (num % i == 0) {
                isPrime = false;
                break;
            }
        }
    }

    if (isPrime)
        cout << "Prime number";
    else
        cout << "Not a prime number";

    return 0;
}
```

Q10. Program to display first 15 even numbers

```
#include <iostream>
using namespace std;

int main() {
    for (int i = 1; i <= 15; i++) {
        cout << i * 2 << endl;
    }
    return 0;
}
```

Q11. Program to display table of a number

```
#include <iostream>
using namespace std;

int main() {
    int num;

    cout << "Enter a number: ";
    cin >> num;

    for (int i = 1; i <= 10; i++) {
        cout << num << " * " << i << " = " << num * i << endl;
    }

    return 0;
}
```

Q12. Program using do-while loop (input between 0 and 15)

```
#include <iostream>
using namespace std;

int main() {
    int num, sum = 0;
```

```
    do {
        cout << "Enter a number between 0 and 15: ";
        cin >> num;

        if (num >= 0 && num <= 15)
            sum += num;

    } while (num < 0 || num > 15);

    cout << "Sum of valid numbers = " << sum;

    return 0;
}
```

Q13. Program to produce pattern

Output:

```
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
```

Program:

```
#include <iostream>
using namespace std;

int main() {
    for (int i = 0; i <= 5; i++) {
        for (int j = 0; j <= i; j++) {
            cout << j << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Q14. Program to produce output

Output:

```
0   1
1   2
2   4
3   8
4   16
5   32
6   64
```

```
#include <iostream>
using namespace std;

int main() {
   for (int i = 0; i <= 6; i++) {
      cout << i << "\t" << (1 << i) << endl;
   }
   return 0;
}
```

GOVT
ISLAMIA GRADUATE COLLEGE
KASUR

GOVT
ISLAMIA GRADUATE COLLEGE
KASUR