

## According to Pairing scheme 2025

### 2nd Year pairing scheme 2025

Q. No. 1 (MCQs)	
Chap 1	1
Chap 2	1
Chap 3	1
Chap 4	1
Chap 5	1
Chap 6	1
Chap 7	0
Chap 8	2
Chap 9	1
Chap 10	1
Chap 11	1
Chap 12	2
Chap 13	1
Chap 14	1
Total	15

Q. No. 2 (Short Questions)	
Chap 1	3
Chap 2	2
Chap 3	0
Chap 4	2
Chap 5	2
Chap 6	0
Chap 7	0

Q. No. 3 (Short Questions)	
Chap 8	3
Chap 10	4 (T, T, E, O)
Chap 14	2

Q. No. 4 (Short Questions)	
Chap 11	3 (T, T, E)
Chap 12	3 (T, E, O)
Chap 13	3 (T, T, T)

Q. No. 5	Chap 3
Q. No. 6	Chap 6 or 7

Q. No. 7	Chap 9
Q. No. 8	Chap 12 (Program)
Q. No. 9	Chap 11 (Program)

* T Means Theory
* E Means Error
* O means Output

### All long Questions (Important)

#### Chapter 3

1. What is data modeling. Discuss ingredients of data modeling. (Book Pg. 23)
2. Explain ER diagram with an example. (Book Pg. 26)
3. Explain Conceptual (also called logical) database design. (Book Pg. 28)
4. Explain physical database design / Explain components of physical database design / Explain Data distribution strategies. (same answer on (Book Pg. 29)

#### Chapter 9

1. Define identifier. Explain its types (Book Pg. 145)
2. Define variable. Explain variable declaration and initialization. (Book Pg. 146)
3. Define variable. Explain rules for naming variables. (Book Pg. 148)
4. Explain data types for integers. (Book Pg. 150)
5. Explain data types for floating point numbers. (Book Pg. 150)
6. Explain data types for characters. (Book Pg. 151)
7. Explain logical operators. (Book Pg. 153)
8. Explain increment decrement operators. (Book Pg. 155)
9. Define comment. Explain its types. (Book Pg. 159)

#### Chapter 11 + 12

1. Do program from these two chapters (Example and Exercise)

## Chapter 3

### Q1. What is Data Modeling? Discuss Its Main Components with Examples.

**Data modeling** is the process of identifying important data items and showing how they are related. It helps in designing the structure of a database clearly. Data modeling is often done using the **Entity-Relationship (ER) model**.

#### Entity

An entity is anything that exists and has importance in the system. It can be a person, place, object, or idea. Examples of entities include STUDENT, TEACHER, CAR, or CLASS. In ER diagrams, an entity is shown using a **rectangle**.

#### Attribute

Attributes describe the features or properties of an entity. For example, the STUDENT entity may have attributes like Name, Roll No, and Address. In an ER diagram, attributes are shown using **oval** shapes connected to the entity.

#### Relationship

A relationship shows how two or more entities are linked. For example, a BOOKSTORE may **sell** or **stock** BOOKS. In an ER diagram, relationships are shown using **diamond** shapes placed between the related entities.

---

### Q2. What is an ER Diagram? Explain with an Example.

An **ER diagram** is a drawing that shows the structure of a database. It explains how data is stored and how different data items are related. ER diagrams are used to design the database system before building it.

#### Rectangle: Entity

Entities are shown using **rectangles** in the ER diagram. Each entity stands for a real-world object like a STUDENT, TEACHER, or CAR.

#### Oval: Attribute

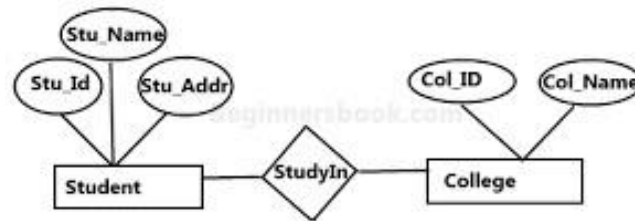
Attributes are the details or features of an entity. They are shown using **ovals** and are connected to the related entity using a line.

## Diamond: Relationship

Relationships are shown using **diamond** shapes. They describe how two or more entities are related. The name of the relationship is written inside the diamond.

## Lines: Connectors

Lines are used to connect entities with their attributes and relationships. They help show how everything is linked.



Sample E-R Diagram

## Example – College Enrollment System

In a college enrollment system, the main entities are STUDENT and COLLEGE. A STUDENT enrolls in a COLLEGE. The attributes of the STUDENT include stuID, stuName, and stuAddress, while the attributes of the COLLEGE include colID and colName. These actions are represented as relationships in the ER diagram, where a STUDENT is associated with a COLLEGE.

---

## Q3. What is Conceptual or Logical Database Design?

**Conceptual database design**, also called **logical database design**, is the step where the full structure of the database is planned. It helps to organize data properly before it is stored in the system.

## Representing Entities

In this step, each entity from the ER diagram becomes a table in the database. The name of the entity becomes the table name. The main field (like Roll No for STUDENT) becomes the **primary key**. Other fields are called **non-key attributes**.

## Representing Relationships

Relationships are added in two ways. One way is to use a foreign key, which is the primary key of another table. For example, in a college system, the **STUDENT** table can include **colID** as a foreign key, which links to the **COLLEGE** table, showing which college the student belongs to.

## Merging Relations

Sometimes, there are repeated or similar tables. These are merged into one to remove extra data. For example, if we have separate tables for **STUDENT** and **STUDENT\_ADDRESS**, we can merge them into one table to avoid repeating address information. This process is called **view integration**.

## Normalization

Normalization is used to make the data structure simple and error-free. It removes extra data and solves problems like update errors. For example, instead of storing a student's address in multiple places, we store it in one table and link it to the student's record, making the database easier to manage.

---

## Q4. What is Physical Database Design?

**Physical database design** is the step where the logical plan is changed into a working system. This design controls how data is stored on the computer and how fast and safe it is to use.

### Data Volume and Usage

This step checks how big the database will be and how often it will be used. It helps decide how much storage is needed and how to set up the system.

### Data Distribution Strategy

This decides where the data will be stored, especially in systems that work over a network.

- **Centralized:** All data is stored in one place.
- **Partitioned:** Data is divided and stored in different locations.
- **Replicated:** Full copies of the data are kept at many locations.
- **Hybrid:** Important data is copied to many places; other data is kept in one place.

### File Organization

This means how data is saved on the storage device. The designer chooses the best method based on the hardware, software, and user needs.

## Indexes

Indexes help find data faster. They work like a search guide. Indexes can be made for main fields like the primary key or for other fields too.

## Integrity Constraints

These are rules to make sure that data is correct and reliable. For example, no two students can have the same ID. These rules protect the quality of the data.

Certainly! Here's a **combined long question** that explains all four **data distribution strategies** (Centralized, Partitioned, Replicated, and Hybrid) along with their **advantages and disadvantages**, written in one clear and simple answer.

---

### Q5. What Are Data Distribution Strategies? Explain Centralized, Partitioned, Replicated, and Hybrid Approaches with Their Advantages and Disadvantages.

In a distributed database system, data can be stored in different ways depending on the needs of users and the network. These methods are called **data distribution strategies**. They help in deciding where the data should be kept so that it is available, fast to access, and safe. The main types of data distribution strategies are: **Centralized, Partitioned, Replicated, and Hybrid**.

#### Centralized Data Distribution

In centralized distribution, **all the data is stored at one location or server**. Every user connects to this single place to access or update the data. This method is usually used in small organizations or when all users work in the same building.

#### Advantages:

- Easy to manage because the data is in one place.
- Simple backup and security management.
- No chance of data conflict since there is only one copy.

#### Disadvantages:

- If the central system fails, the entire database stops working.
- Remote users may experience slow access.
- High communication cost for distant users.

## Partitioned Data Distribution

In this method, the database is **divided into smaller parts**, called **partitions**, and each part is stored at a **different location**. Each location stores only the data it needs.

### Advantages:

- Data is closer to users, so access becomes faster.
- Reduces the load on the main server.
- Can improve speed and system performance.

### Disadvantages:

- Difficult to manage many locations.
- Searching across all partitions takes more time.
- Needs careful planning to divide the data properly.

## Replicated Data Distribution

In replicated distribution, **a full copy of the database** is stored at **multiple sites**. Every site has the same data.

### Advantages:

- Data remains available even if one site fails.
- Faster access for local users.
- Increases system reliability and safety.

### Disadvantages:

- Hard to update all copies at the same time.
- Risk of different data at different places if updates fail.
- Requires more storage space.

## Hybrid Data Distribution

Hybrid distribution is a **mix of centralized, partitioned, and replicated strategies**. In this system, **important (critical) data is copied to many places**, and **non-critical data is kept at one location**.

### Advantages:

- Balances speed, cost, and storage.
- Important data is always available to users.

- Offers flexibility and better performance.

**Disadvantages:**

- More complex to manage.
- Needs planning to decide what data is critical.
- If non-critical data is missing, it may cause problems.

**Conclusion**

These strategies are chosen based on the size of the organization, number of users, network structure, and the importance of data availability. Each has its own benefits and limitations, and choosing the right one helps in building a fast and reliable database system.

## Chapter 9

---

### 1. Define identifier. Explain its types.

An **identifier** is the name given to a variable, constant, function, or label in a program. It is used to identify the name of a memory location or function in a meaningful way.

#### Types of Identifiers:

##### 1. Standard Identifiers:

These are pre-defined identifiers that have a special meaning in C language. They are part of the C library and are used to perform specific functions. Standard identifiers cannot be changed or used for other purposes in the program **Examples:** printf, scanf

##### 2. User-defined Identifiers:

These are created by the programmer to store data or program results. They are used for variables, functions, and other user-defined elements. These identifiers can be named as per the programmer's choice, as long as they follow certain naming rules in C. **Examples:** my\_name, marks, age

##### 3. Rules for Naming Identifiers:

- Identifiers must start with a letter (a-z, A-Z) or an underscore (\_), followed by letters, digits (0-9), or underscores.
  - Identifiers cannot be the same as keywords (like int, return, if, etc.).
  - C is case-sensitive, so my\_name and MY\_NAME would be considered two different identifiers.
-



## 2. Define variable. Explain variable declaration and initialization.

A **variable** is a named memory location used to store input data and the results of computations during program execution. Its value can be changed during execution, and new values replace old ones.

### Variable Declaration:

It is the process of specifying the variable name and its data type. The compiler uses this to allocate memory. **Syntax:** data\_type variable\_name; **Example:** int marks;

### Variable Initialization:

Assigning a value to a variable at the time of declaration is called initialization.

#### Syntax:

data\_type variable\_name = value; **Example:**

int n = 100;

If not initialized, variables may hold garbage values and produce errors.

---

## 3. Define variable. Explain rules for naming variables.

A **variable** is a named memory cell that holds data that may change during the execution of a program.

### Rules for Naming Variables:

- The first character must be a letter or underscore \_.
  - Can contain letters, digits, and underscores.
  - Cannot contain spaces or special characters like @, #, %.
  - Cannot use reserved keywords like int, double.
  - Variables are case-sensitive.
  - Should be meaningful and descriptive.
  - Follow naming conventions like InterestRate or my\_name.
-

#### 4. Explain data types for integers

In C language, integer data types are used to store whole numbers that do not have decimal points. These numbers can be either positive or negative. C provides different types of integer data types to store numbers of different ranges.

##### Types of Integer Data Types in C: int

###### (Integer):

This data type is used to store normal whole numbers. It uses 2 bytes of memory and stores values from  $-32,768$  to  $+32,767$ .

###### short int (or short):

This is also used to store whole numbers. It uses 2 bytes of memory, and its range is the same as int, which is  $-32,768$  to  $+32,767$  ( $-2^{15}$  to  $2^{15} - 1$ ).

This data type is used to store large integer values. It uses 4 bytes of memory and can store values from  $-2,147,483,648$  to  $+2,147,483,647$  ( $-2^{31}$  to  $2^{31} - 1$ ).

**unsigned int:**  
This type stores only positive whole numbers. It uses 2 bytes of memory and stores values from 0 to  $65,535$  ( $0$  to  $2^{16} - 1$ ).

###### unsigned long int:

This type is used to store large positive numbers. It uses 4 bytes of memory and stores values from 0 to  $4,294,967,295$  ( $0$  to  $2^{32} - 1$ ).

---

#### 5. Explain data types for floating point numbers

Floating point data types are used to store numbers that have decimal points or fractional parts. These are also called real numbers. In C language, there are different floating point data types to store numbers with different sizes and precision.

##### Types of Floating Point Data Types in C:

###### float:

This data type is used to store real numbers with small precision. It uses 4 bytes of memory and stores values from  $10^{-38}$  to  $10^{38}$ . It gives up to 6 digits after the decimal point.

###### double:

This is used for storing large real numbers with higher precision. It uses 8 bytes of memory and can store values from  $10^{-308}$  to  $10^{308}$ . It gives up to 15 digits after the decimal point.

**long double:**

This data type is used for storing very large real numbers with very high precision. It uses 10 bytes of memory and can store values from  $10^{-4932}$  to  $10^{4932}$ . It gives up to 19 digits after the decimal point.

---

## **6. Explain data types for characters**

The char data type is used to store a single character like a letter, digit, or symbol. It takes 1 byte of memory and stores the ASCII value of the character. For example, the character 'A' is stored as 65 and 'a' as 97. Even numbers like '6' are stored using their ASCII codes, such as 54.

**Features of char:**

The char type stores one character and uses 1 byte of memory. Characters are written in single quotes like 'a', '5', or '\$'.

**Signed and Unsigned char:**

By default, char is unsigned. An **unsigned char** can store values from 0 to 255. A **signed char** can store values from -128 to +127.

Since characters are stored as numbers (ASCII values), we can also do arithmetic operations on them, like adding or comparing characters.

## Chapter 11 Exercise Solved Programs

### Example 1: Calculate Square root of a positive number

```
#include <stdio.h>
#include <math.h>

int main() {
    int number;

    // Input positive number
    printf("Enter a positive number: ");
    scanf("%d", &number);

    // Check if the number is positive
    if (number < 0) {        printf("Please enter a
positive number.\n");
        return 1; // Exit with an error code
    }

    // Calculate and display the square root
    float squareRoot = sqrt(number);
    printf("Square root of %d: %.2f \n", number, squareRoot);
}
```

---

### Example 2: Accepts three numbers and display largest number.

```
#include <stdio.h> int
main() {
    // Input three numbers    float
    num1, num2, num3;
    printf("Enter three numbers: ");
    scanf("%f %f %f", &num1, &num2, &num3);

    // Check and display the largest number    if
    (num1 >= num2 && num1 >= num3) {
        printf("The largest number is %.2f\n", num1);    }
    else if (num2 >= num1 && num2 >= num3) {
        printf("The largest number is %.2f\n", num2);
    } else {
        printf("The largest number is %.2f\n", num3);
    }
}
```

---

---

**Example 3: Accepts a number from user and determine whether positive, negative or zero.**

```
#include <stdio.h>
int
main() {
    // Input three numbers    float
    num1, num2, num3;
    printf("Enter three numbers: ");
    scanf("%f %f %f", &num1, &num2, &num3);

    // Check and display the largest number    if
    (num1 >= num2 && num1 >= num3) {
        printf("The largest number is %.2f\n", num1);    }
    else if (num2 >= num1 && num2 >= num3) {
        printf("The largest number is %.2f\n", num2);
    } else {
        printf("The largest number is %.2f\n", num3);
    }
}
```

---

---

**Example 4: Accepts a character from user and determine whether it is vowel or not.**

```
int main() { // Input
    a character
    char ch;
    printf("Enter a character: ");
    scanf(" %c", &ch);
    // Check and display whether the character is a vowel or not
    switch (ch) {    case 'a':    case 'A':    case 'e':    case
'E':    case 'i':    case 'I':    case 'o':    case 'O':
case 'u':    case 'U':
        printf("The character '%c' is a vowel.\n", ch);
    break;    default:
        printf("The character '%c' is not a vowel.\n", ch);
    }
}
```

---

---

## Chapter 11 Exercise Solved Programs

**Question 8: write a program that Compute the area of a square or a triangle ( Ask user to enter S for square and T for Triangle)** #include <stdio.h> int main() { char figureType;

```
    printf("Enter the first character of the figure name (S or T): ");
    scanf(" %c", &figureType);
```

```
    if (figureType == 'S' || figureType == 's') {
        float side, area;
        printf("Enter the side length of the square: ");
        scanf("%f", &side);    area = side * side;
        printf("Area of the square: %f\n", area);    } else if
        (figureType == 'T' || figureType == 't') {    float
        base, height, area;
        printf("Enter the base length of the triangle: ");
        scanf("%f", &base);
        printf("Enter the height of the triangle: ");
        scanf("%f", &height);    area = 0.5 * base *
        height;
        printf("Area of the triangle: %f\n", area);
    } else {
        printf("Invalid figure type.\n");
    }
}
```

---

**// Question 9: Check if a year is a leap year**

```
#include <stdio.h> int
main() { int year;
printf("\nEnter a year: ");
scanf("%d", &year);

    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
printf("Leap year\n");
    } else {
        printf("Not a leap year\n");
    }
}
```

---

**// Question 10: Display a message based on temperature**

```
#include <stdio.h> int main() {
float temperature;
printf("\nEnter temperature: ");
scanf("%f", &temperature);

if (temperature > 35) {
printf("Hot day\n");
} else if (temperature >= 25 && temperature <= 35) {
printf("Pleasant day\n");
} else {
printf("Cool day\n");
}
}
```

---

**// Question 11: Calculate percentage and display grade**

```
#include <stdio.h>
int main() { int
obtainedMarks;
printf("\nEnter obtained marks (out of 1100): ");
scanf("%d", &obtainedMarks);

float percentage = (float)obtainedMarks / 1100 * 100;

if (percentage >= 80) {
printf("Grade: A+\n"); } else
if (percentage >= 70) {
printf("Grade: A\n"); } else
if (percentage >= 60) {
printf("Grade: B\n"); } else
if (percentage >= 50) {
printf("Grade: C\n"); } else
if (percentage >= 40) {
printf("Grade: D\n");
} else {
printf("Grade: F\n");
}
}
```

---

**// Question 12: Perform arithmetic operation based on user choice**

```
#include <stdio.h> int main() {
int choice; float num1, num2,
result; printf("\nEnter two
numbers: ");
scanf("%f %f", &num1, &num2);

printf("Enter your choice (1: Sum, 2: Subtract, 3: Multiply, 4: Divide): ");
scanf("%d", &choice);

switch (choice) {
case 1:
result = num1 + num2;
printf("Sum: %f\n", result);
break;
case 2:
result = num1 - num2;
printf("Difference: %f\n", result);
break;
case 3:
result = num1 * num2;
printf("Product: %f\n", result);
break;
case 4:
if (num2 != 0) { result
= num1 / num2;
printf("Quotient: %f\n", result);
} else {
printf("Error: Division by zero\n");
}
break;
default:
printf("Wrong choice\n");
}

}
```



## Chapter 12 Example Programs

### Example 3

Write a program that will print asterisks (\*) according to the pattern shown in the fig. 12.2.

```
#include <stdio.h>
```

```
void main(void)
```

```
{
```

```
    int inner;
```

```
    for(int outer=7; outer>=1; outer--)
```

```
    {
```

```
        inner = 1;
```

```
        while( inner <= outer)
```

```
        {
```

```
            printf("*");
```

```
            inner++;
```

```
        }
```

```
        printf("\n");
```

```
    }
```

```
}
```

```
*****
```

```
*****
```

```
*****
```

```
****
```

```
***
```

```
**
```

```
*
```

Fig. 12.2 asterisks pattern

**Example 4** Write a program to find the average marks of the students in a class.

```
#include <stdio.h>

void main(void)
{
    int sum = 0, marks, total_students = 0;
    float average;

    do{
        printf("Enter marks of the student (or any -ve  
number to quit) >");
        scanf("%d", &marks);
        if (marks >= 0)
        {
            total_students++;
            sum += marks;
        }
    } while(marks >= 0);

    if (total_students > 0)
    {
        average = sum / (float)total_students;
        printf("The average marks of the class are:  
%f\n", average);
    }
    else
        printf("Please enter the marks of at least one  
student to calculate average\n");
}
```

**Example 5**

Write a program to calculate the square root of a positive number. Also handle negative numbers properly.

```
#include <math.h>
#include <stdio.h>

void main()
{
    float num;

positive:
    printf("Please Enter a positive number: ");
    scanf("%f", &num);

    if (num < 0)
        goto positive;
    else
        printf("Square root of %0.2f is %0.2f", num,
            sqrt(num));
}
```

## Chapter 12 Exercise Programs

**9. Write a program that inputs a number and displays the message "Prime number" if it is a prime number, otherwise displays "Not a prime number".**

```
#include <stdio.h>
```

```
int main() {

    int num, i, isPrime = 1;

    printf("Enter a number: ");

    scanf("%d", &num);

    if (num <= 1) {

        isPrime = 0;

    } else {

        for (i = 2; i < num; i++) {

            if (num % i == 0) {

                isPrime = 0;

                break;

            }

        }

    }

}
```

```
    }  
    }  
}  
if (isPrime == 1)  
    printf("Prime number\n");  
else  
    printf("Not a prime number\n");  
}
```

10. Write a program that displays the first 15 even numbers.

```
#include <stdio.h>

int main() {

    for (int i = 1; i <= 15; i++) {

        printf("%d\n", i * 2);

    }

}
```

11. Write a program that inputs a number, and displays its table according to the following format:

Suppose the number entered is 5, the output will be as follows:

```
5*1=5
5*2=10
5*3=15
.....
5*10=50
```

```
#include <stdio.h>

int main() {

    int num, i;

    printf("Enter a number: ");

    scanf("%d", &num);

    for (i = 1; i <= 10; i++) {

        printf("%d * %d = %d\n", num, i, num * i);

    }

}
```

12. Write a program using do-while loop that repeatedly prompts for and takes input until a value in the range 0 through 15 inclusive is input. The program should add all the values before exiting the loop and displays their sum at the end.

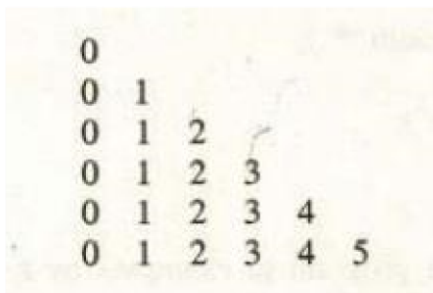
```
#include <stdio.h>

int main() {
    int num, sum = 0;
    do {
        printf("Enter a number (0-15): ");
        scanf("%d", &num);

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

    printf("Sum of entered numbers: %d\n", sum);
}
```

13. Write a program that produces the following output



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

```
#include <stdio.h>

int main() {
    int i, j;
    for (i = 0; i <= 5; i++) { // Loop for rows
        for (j = 0; j <= i; j++) { // Loop for numbers in each row
            printf("%d ", j);
        }
        printf("\n"); // Move to the next line after each row
    }
}
```

14. Write a program that produces the following output:

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

```
#include <stdio.h>

int main() {
    int i, value = 1;
    for (i = 0; i <= 6; i++) {
        printf("%d %d\n", i, value);
        value *= 2; // Multiply the value by 2 for the next iteration
    }
}
```

## Board Sample Paper

### (Objective Type)

**NOTE:** Write your **Name** in the space provided. Over-writing, Cutting, Erasing, using lead pencil will result in loss of marks.

**Q.1:** You have four choices for each objective type question as A, B, C and D. The choice which you think is correct; fill that circle in front of that question number. Use marker or pen to fill the circle. Cutting or filling two or more circles will result in zero marks in that question. Write the letter A, B, C or D in the column (Write correct option) against each question. If there is a contradiction in the bubble and hand written answer bubble option will be considered correct. (15)

1	How many are the layout of report? a) 2                      b) 3                      c) 4                      d) 5
2	Row in the Table is also known as: a) Relation              b) Tuple                  c) record                  d) field
3	The output of a query is: a) Table                  b) form                  c) report                  d) macro
4	A loop within another loop is called: a) finite loop            b) infinite loop          c) nested loop            d) sentinel loop
5	Two or more attributes with different names but same meaning create a problem know as: a) homonyms            b) aliases                c) synonyms              d) alternate attribute
6	In ERD model, the relationship between two entities is represented by a: a) diamond symbol    b) rectangle              c) oval symbol            d) arrow
7	Which form of dependency is removed in 3NF? a) functional            b) non functional        c) Transitive              d) Associative
8	An ampersand (&) before the name of a variable denotes: a) data type              b) address                c) value                    d) format
9	What does a compound condition used to join two conditions? a) Relational Operator    b) Logical Operator      c) Assignment operator    d) Arithmetic Operator
10	Which programming structure executes program in an order: a) sequence              b) relation                c) decision                d) repetition
11	The scope of the variable refer to its: a) length                  b) name                    c) accessibility            d) data type
12	One execution of loop is known as: a) duration                b) test                      c) iteration                d) Continuity
13	Formal arguments are also called: a) Actual argument      b) dummy argument      c) original arguments      d) referenced arguments
14	Which of the following character is used to mark the end of string: a) \o                      b) \r                      c) \a                      d) \n
15	_____ function is used to write a character to a file: a) fgetc()                b) fputs()                c) fputc()                d) fgets()



**Note: Write same question number on answer sheet as given in question paper.**

**(Section I)**

**Q.2: Write short answers to any SIX (6) questions. (12)**

- i. What is the purpose of Master file?
- ii. Differentiate between primary and secondary key.
- iii. Write the responsibilities of Database administrator (DBA).
- iv. Differentiate between cardinality and modality.
- v. Define referential integrity.
- vi. What is the purpose of toolbar in MS Access?
- vii. What is meant by Degree of a Relation?
- viii. State condition for 1NF?
- ix. When does deletion anomaly occur?

**Q.3: Write short answers to any SIX (6) questions. (12)**

- i. Differentiate between linker and loader.
- ii. What are preprocessor directives?
- iii. Discuss logical error.
- iv. How does compiler and interpreter work?
- v. Define Expression.
- vi. `int j=10; int i=-j;` what will be the value of i and j after execution.
- vii. Discuss numeric and character constants with example.
- viii. How do you open a file with name myfile in read mode?
- ix. Define binary stream.

**Q.4: Write short answers to any SIX (6) questions. (12)**

**i. Write output**

```
int main() {
    int i=4;
    for (int j = 0; j < 5; j++) {
        printf("\n%d%d", j,i);
    }
}
```

**iii. Find error**

```
int main() {
    i = 0;
    while (i < 5);
    {
        printf("%d ", i);
        i++;
    }
}
```

- v. Write syntax of goto statement.
- vi. Draw flowchart of if else statement
- vii. Write syntax of do while loop.
- viii. What are escape sequence. Given an example.
- ix. Write syntax to define a function.

**ii. Write output**

```
int main(){
    int x=20;
    if(x!=10)
        printf("Hello");
    else
        printf("World");
}
```

**iv. Find error**

```
#include <stdio.h>
int main() {
    int count = getch();
    do {
        printf("Count: %d\n", count);
        count++;
    } while (count < 5)
}
```

**(Section II)**

**(MS ACCESS)**

**Note: Attempt any ONE question. (8×1=8)**

**Q No. 5: Discuss different data distribution strategies. (8)**

**Q No. 6: Explain any four data types available in MS Access. (8)**

**(C-Language)**

**Note: Attempt any TWO questions. (8×2=16)**

**Q No. 6: Define variable explain rules for naming variable. (8)**

**Q No. 8: Write a program that inputs a character from the user and check whether it is a vowel or constant. (8)**

**Q No. 9: Write a program that inputs a number and displays its Table using loop. (8)**