Manipur Board 2026 Class 12 Computer Science Question Paper with Solutions(Memory Based)

Time Allowed :3 Hour	Maximum Marks :70	Total Questions :36

General Instructions

Read the following instructions very carefully and strictly follow them:

- The exam lasts 3 hours.
- The theory paper is typically worth 70 marks, with the remaining 30 marks allocated to the practical examination.
- The paper is divided into three sections:
  Section A: 10 questions of 1 mark each, requiring brief answers (approx. 20–30 words).
  Section B: 5 questions of 3 marks each, requiring longer answers (approx. 100–150 words).
  Section C: 5 questions of 5 marks each (approx. 250–300 words), often with internal choice options.
- All sections are generally compulsory, but internal choices are provided within specific questions.
- For any discrepancy in questions, the English version is considered final (except for language-specific papers).
- Students are granted an extra 15 minutes specifically to read the question paper and plan their answers before writing begins.

1. Write a Python function to count the number of vowels in a given text file named `story.txt`.

Solution:

Concept:
In Python, files can be read using built-in file handling functions such as `open()`, `read()`, and `close()`. To count vowels in a text file, the program must perform the following tasks:

- Open the file `story.txt` in read mode.
- Read the content of the file.
- Check each character to determine whether it is a vowel.
- Count the total number of vowels present.

The vowels in the English alphabet are:

$$a, e, i, o, u$$

Both uppercase and lowercase vowels should be considered while counting.

Step 1: Define a function to perform the task.

A Python function is defined using the `def` keyword. The function will read the file and count vowels.

```
def count_vowels():
    vowels = "aeiouAEIOU"
    count = 0
```

Step 2: Open and read the text file.

The file `story.txt` is opened in read mode using the `open()` function.

```
    file = open("story.txt", "r")
    text = file.read()
```

This reads the entire content of the file into a variable called `text`.

Step 3: Check each character for vowels.

Each character in the text is examined using a loop. If the character is present in the vowel list, the counter increases.

```
    for ch in text:
        if ch in vowels:
            count += 1
```

Step 4: Display the result and close the file.

Finally, print the total number of vowels and close the file.

```
    print("Number of vowels:", count)
    file.close()
```

Complete Python Function:

```
def count_vowels():
    vowels = "aeiouAEIOU"
    count = 0

    file = open("story.txt", "r")
    text = file.read()

    for ch in text:
        if ch in vowels:
            count += 1

    print("Number of vowels:", count)
    file.close()
```

Step 5: Calling the function.

To execute the function, it must be called in the program.

```
count_vowels()
```

Quick Tip

Useful Python file handling steps:

- **open()** → Opens a file
- **read()** → Reads the contents of a file
- **close()** → Closes the file after use

Always close files after reading them to free system resources.

---

2. Explain the difference between `list.append()` and `list.extend()` with a code example.

Solution:

Concept:
In Python, lists are mutable data structures that allow elements to be added, removed, or modified. Two commonly used list methods for adding elements are:

- `list.append()`
- `list.extend()`

Although both methods add elements to a list, they behave differently when adding another list or iterable.
Step 1: Understanding `list.append()`.
The `append()` method adds **a single element** to the end of the list. If a list is appended, the entire list becomes a **single element** inside the original list.

```
numbers = [1, 2, 3]
numbers.append([4, 5])

print(numbers)
```

Output:

```
[1, 2, 3, [4, 5]]
```

Here, the list `[4,5]` is added as **one element**.
Step 2: Understanding `list.extend()`.
The `extend()` method adds **each element of an iterable** (such as a list, tuple, or string) individually to the list.

```
numbers = [1, 2, 3]
numbers.extend([4, 5])

print(numbers)
```

Output:

```
[1, 2, 3, 4, 5]
```

Here, the elements `4` and `5` are added individually to the list.
Step 3: Key differences between append() and extend().

Feature	append()	extend()
Elements added	Single element	Multiple elements
If a list is added	Added as one element	Elements added individually
Resulting structure	Nested list possible	Flat list structure

Step 4: Combined Example

```
list1 = [10, 20, 30]

list1.append([40, 50])
print("After append:", list1)

list2 = [10, 20, 30]
list2.extend([40, 50])
print("After extend:", list2)
```

Output:

```
After append: [10, 20, 30, [40, 50]]
After extend: [10, 20, 30, 40, 50]
```

Quick Tip

Easy way to remember:

- **append()** → Adds one element (even if it is a list)
- **extend()** → Adds elements of an iterable individually

Shortcut memory trick:
append = add one item
extend = expand the list

---

3. Write a program to implement a Stack using a Python list with push and pop operations.

Solution:

Concept:
A **Stack** is a linear data structure that follows the principle of **LIFO (Last In, First Out)**. This means the last element inserted into the stack is the first one to be removed.
Common operations performed on a stack include:

- **Push** – Insert an element onto the stack

- **Pop** – Remove the top element from the stack
- **Peek** – View the top element without removing it

In Python, a stack can easily be implemented using a **list**, since lists support dynamic insertion and deletion of elements.
Step 1: Create an empty list to represent the stack.
An empty list will act as the stack container.

```
stack = []
```

Step 2: Define the push operation.
The push operation adds an element to the top of the stack. In Python lists, this can be done using the `append()` method.

```
def push(item):
    stack.append(item)
    print(item, "pushed to stack")
```

Step 3: Define the pop operation.
The pop operation removes the last element from the stack. Python lists provide the `pop()` method to remove the last element.

```
def pop():
    if len(stack) == 0:
        print("Stack is empty")
    else:
        item = stack.pop()
        print(item, "popped from stack")
```

Step 4: Complete Python Program

```
# Stack implementation using list

stack = []

def push(item):
    stack.append(item)
    print(item, "pushed to stack")

def pop():
    if len(stack) == 0:
        print("Stack is empty")
    else:
        item = stack.pop()
        print(item, "popped from stack")

# Example usage
push(10)
push(20)
```

```
push(30)

print("Current Stack:", stack)

pop()
pop()

print("Stack after pop operations:", stack)
```

Step 5: Expected Output

```
10 pushed to stack
20 pushed to stack
30 pushed to stack
Current Stack: [10, 20, 30]
30 popped from stack
20 popped from stack
Stack after pop operations: [10]
```

Quick Tip

Important stack concepts:

- **Stack principle:** LIFO (Last In First Out)
- **Push operation:** Use `append()`
- **Pop operation:** Use `pop()`

Python lists provide a simple and efficient way to implement stacks.

---

4. What is the difference between a Hub and a Switch in computer networking?

Solution:

Concept:
In computer networking, both **Hub** and **Switch** are networking devices used to connect multiple computers within a **Local Area Network (LAN)**. They allow devices to communicate with each other and share data.
However, these two devices differ in terms of **working principle, efficiency, and data transmission method**. A hub is a simpler device that broadcasts data to all connected devices, whereas a switch is more intelligent and sends data only to the intended device.
Step 1: Understanding Hub.
A **Hub** is a basic networking device that operates at the **Physical Layer (Layer 1)** of the OSI model.

- It connects multiple computers in a network.

- When a hub receives data from one device, it **broadcasts the data to all connected devices**.
- Every device checks whether the data is intended for it.
- This can cause unnecessary traffic in the network.

Because of this broadcasting behavior, hubs are generally slower and less efficient.
Step 2: Understanding Switch.
A **Switch** is a more advanced networking device that operates at the **Data Link Layer (Layer 2)** of the OSI model.

- It connects multiple devices in a network similar to a hub.
- A switch maintains a **MAC address table**.
- It sends data only to the **specific device** for which the data is intended.
- This reduces unnecessary network traffic and improves efficiency.

Therefore, switches are faster and more efficient than hubs.
Step 3: Key differences between Hub and Switch.

Feature	Hub	Switch
OSI Layer	Physical Layer	Data Link Layer
Data Transmission	Broadcasts to all devices	Sends to specific device
Efficiency	Low	High
Traffic Control	No traffic management	Reduces network traffic
Speed	Slower	Faster

Step 4: Conclusion.
Both devices connect multiple computers in a network, but:

- A **Hub** broadcasts data to all devices.
- A **Switch** sends data only to the intended device using MAC addresses.

Therefore, switches are more efficient and widely used in modern computer networks.

Quick Tip

Simple way to remember:

- **Hub** → Broadcasts data to all devices
- **Switch** → Sends data to the correct device

Hub = Less intelligent
Switch = Smart device

---

5. Define IP Address and explain the difference between IPv4 and IPv6.

Solution:

Concept:
An **IP Address (Internet Protocol Address)** is a unique numerical identifier assigned to each device connected to a computer network that uses the Internet Protocol for communication. It allows devices such as computers, smartphones, and servers to identify and communicate with each other over the internet.
An IP address performs two main functions:

- **Host Identification** – Identifies a specific device on a network.
- **Location Addressing** – Indicates the location of the device within the network.

There are two main versions of IP addresses used in networking:

- **IPv4 (Internet Protocol Version 4)**
- **IPv6 (Internet Protocol Version 6)**

Step 1: Understanding IPv4.
IPv4 is the fourth version of the Internet Protocol and is the most widely used addressing system.

- It uses a **32-bit address**.
- The address is written in **decimal format**.
- It consists of **four octets** separated by dots.

Example of an IPv4 address:

$$192.168.1.1$$

IPv4 can provide approximately 2^{32} unique addresses, which equals about **4.3 billion addresses**. Due to the rapid growth of internet devices, IPv4 addresses have become insufficient.
Step 2: Understanding IPv6.
IPv6 was developed to overcome the limitation of IPv4 address exhaustion.

- It uses a **128-bit address**.
- The address is written in **hexadecimal format**.
- It consists of **eight groups** separated by colons.

Example of an IPv6 address:

$$2001 : 0db8 : 85a3 : 0000 : 0000 : 8a2e : 0370 : 7334$$

IPv6 can generate approximately 2^{128} unique addresses, which is an extremely large number capable of supporting future internet growth.
Step 3: Key differences between IPv4 and IPv6.

Feature	IPv4	IPv6
Address Length	32-bit	128-bit
Address Format	Decimal	Hexadecimal
Address Example	192.168.1.1	2001:db8::1
Number of Addresses	About 4.3 billion	Very large (2^{128})
Address Groups	4 groups	8 groups

Step 4: Conclusion.
An IP address uniquely identifies a device on a network and enables communication over the internet. IPv4 is the older and widely used addressing system, while IPv6 is the newer version designed to provide a much larger address space and improved network efficiency.

Quick Tip

Important facts to remember:

- **IPv4** → 32-bit address, decimal format
- **IPv6** → 128-bit address, hexadecimal format
- IPv6 was developed to solve the **IPv4 address shortage**.

Example:
IPv4 → `192.168.1.1`
IPv6 → `2001:db8::1`

---

6. Write the SQL command to create a table named `STUDENT` with columns `RollNo`, `Name`, and `Marks`.

Solution:

Concept:
In SQL, a table is created using the **CREATE TABLE** command. This command defines the structure of the table, including the column names and their data types.
A table consists of rows and columns where:

- **Columns** represent attributes or fields.
- **Rows** represent individual records.

When creating a table, each column must be assigned an appropriate **data type** such as `INT`, `VARCHAR`, or `FLOAT`.
Step 1: Understanding the required columns.
The table `STUDENT` should contain the following columns:

- **RollNo** – Stores the student's roll number (integer).
- **Name** – Stores the student's name (string/text).
- **Marks** – Stores the marks obtained by the student (numeric value).

Step 2: SQL command to create the table.

```
CREATE TABLE STUDENT (
    RollNo INT,
    Name VARCHAR(50),
    Marks INT
);
```

Step 3: Explanation of the SQL statement.

- `CREATE TABLE` – Command used to create a new table.
- `STUDENT` – Name of the table being created.
- `RollNo INT` – Defines the roll number column as an integer.
- `Name VARCHAR(50)` – Defines the name column that can store up to 50 characters.
- `Marks INT` – Defines the marks column as an integer.

Step 4: Example table structure after creation.

RollNo	Name	Marks
1	Rahul	85
2	Anjali	92
3	Mohit	78

Quick Tip

Basic SQL table creation syntax:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype
);
```

Always choose appropriate data types such as `INT`, `VARCHAR`, or `DATE`.

---

7. Differentiate between `DELETE` and `DROP` commands in SQL.

Solution:

Concept:
In SQL, both **DELETE** and **DROP** are commands used to remove data, but they operate at different levels of a database.

- **DELETE** removes records (rows) from a table.
- **DROP** removes the entire table structure along with all its data.

Thus, DELETE affects the data stored inside the table, whereas DROP removes the table completely from the database.

Step 1: Understanding the DELETE command.

The **DELETE** command is used to remove one or more rows from an existing table.

- It deletes only the **data (records)**.
- The **table structure remains intact**.
- It can use a **WHERE clause** to delete specific records.

Example:

```
DELETE FROM STUDENT
WHERE RollNo = 5;
```

This command deletes the record where the roll number is 5, but the table `STUDENT` still exists.

Step 2: Understanding the DROP command.

The **DROP** command is used to remove an entire table from the database.

- It deletes **both the data and the table structure**.
- The table cannot be used again unless it is recreated.
- It does **not use a WHERE clause**.

Example:

```
DROP TABLE STUDENT;
```

This command permanently removes the table `STUDENT` from the database.

Step 3: Key differences between DELETE and DROP.

Feature	DELETE	DROP
Type of command	DML (Data Manipulation)	DDL (Data Definition)
Removes	Table records (rows)	Entire table
Table structure	Remains unchanged	Removed completely
WHERE clause	Can be used	Cannot be used
Effect on database	Data deleted only	Table removed permanently

Step 4: Conclusion.

- **DELETE** is used when we want to remove specific records but keep the table.
- **DROP** is used when we want to remove the entire table from the database.

Quick Tip

Simple memory trick:

- **DELETE** → Deletes data inside the table
- **DROP** → Deletes the whole table

Think:
DELETE = Remove rows
DROP = Remove structure

8. What is the purpose of the `ORDER BY` clause in a `SELECT` statement?

Solution:

Concept:
In SQL, the `ORDER BY` clause is used to **sort the result of a query** returned by a `SELECT` statement. It arranges the retrieved records in a specified order based on one or more columns. Sorting can be done in two ways:

- **Ascending order (ASC)** – from smallest to largest or A to Z.
- **Descending order (DESC)** – from largest to smallest or Z to A.

If no order is specified, SQL sorts the results in **ascending order by default**.
Step 1: Basic syntax of ORDER BY.
The general syntax for using the ORDER BY clause is:

```
SELECT column_name
FROM table_name
ORDER BY column_name ASC|DESC;
```

Here:

- `SELECT` retrieves the required data.
- `ORDER BY` sorts the result based on the specified column.
- `ASC` or `DESC` defines the sorting order.

Step 2: Example of ORDER BY in ascending order.

```
SELECT Name, Marks
FROM STUDENT
ORDER BY Marks ASC;
```

This command displays the student records sorted by marks from **lowest to highest**.
Step 3: Example of ORDER BY in descending order.

```
SELECT Name, Marks
FROM STUDENT
ORDER BY Marks DESC;
```

This command displays the student records sorted by marks from **highest to lowest**.
Step 4: Example table output.
If the STUDENT table contains:

RollNo	Name	Marks
1	Rahul	85
2	Anjali	92
3	Mohit	78

After applying:

```
SELECT Name, Marks
FROM STUDENT
ORDER BY Marks DESC;
```

The result will be:

Name	Marks
Anjali	92
Rahul	85
Mohit	78

Quick Tip

Remember:

- **ORDER BY** → Used to sort query results
- **ASC** → Ascending order (default)
- **DESC** → Descending order

Example:
`ORDER BY Marks DESC`

---

9. Explain the role of a Gateway in connecting two dissimilar networks.

Solution:

Concept:
A **Gateway** is a networking device or software that acts as a **connection point between two different networks**. It allows communication between networks that use different protocols, architectures, or data formats.
Gateways are often referred to as **protocol converters** because they translate data from one protocol to another so that devices on different networks can communicate with each other.
For example, a gateway can connect a **local network (LAN)** to the **internet** or connect networks that use completely different communication standards.
Step 1: Understanding the function of a gateway.
The main role of a gateway is to:

- Connect two networks that may use different communication protocols.
- Convert data formats so that devices on both networks can understand each other.
- Manage the flow of data between the two networks.

Thus, it acts as an entry and exit point for data moving between networks.
Step 2: How a gateway works.
When data is sent from one network to another:

- The data first reaches the **gateway**.

- The gateway **translates or converts the protocol**.
- The converted data is then forwarded to the destination network.

This process ensures smooth communication between networks that otherwise could not directly communicate.

Step 3: Example of gateway usage.

A common example is when a **home network connects to the internet**. The router often acts as a gateway by translating the internal network addresses into public internet addresses. Another example is connecting networks using different protocols such as:

- TCP/IP networks
- Legacy communication systems
- Different enterprise networks

Step 4: Importance of a gateway.

Gateways are important because they:

- Enable communication between **different types of networks**.
- Provide **protocol translation**.
- Allow integration of various network technologies.
- Help connect local networks to the global internet.

Step 5: Conclusion.

A gateway acts as an intermediary device that connects and enables communication between two dissimilar networks by translating protocols and managing data exchange.

Quick Tip

Remember the role of networking devices:

- **Hub** → Broadcasts data
- **Switch** → Directs data using MAC address
- **Router** → Connects different networks
- **Gateway** → Connects dissimilar networks and converts protocols

Gateway = Protocol Translator

---

10. What are the advantages of using Optical Fiber over Coaxial Cable for data transmission?

Solution:

Concept:

Optical Fiber and **Coaxial Cable** are two types of transmission media used in computer networks to transfer data between devices. Optical fiber uses **light signals** to transmit data, whereas coaxial cable uses **electrical signals**.
Optical fiber technology provides several advantages over coaxial cables, especially in terms of speed, reliability, and signal quality. Because of these advantages, optical fibers are widely used in modern high-speed communication systems such as broadband internet, telecommunication networks, and data centers.
Step 1: Higher Data Transmission Speed
Optical fiber can transmit data at extremely high speeds because it uses light waves. Light signals travel faster and carry more data compared to electrical signals used in coaxial cables.

- Optical Fiber: Very high bandwidth and speed.
- Coaxial Cable: Lower bandwidth compared to fiber.

Step 2: Greater Bandwidth
Bandwidth refers to the amount of data that can be transmitted in a given time. Optical fiber provides much higher bandwidth than coaxial cable, allowing large amounts of data to be transmitted simultaneously.
This makes optical fiber suitable for applications such as:

- High-speed internet
- Video streaming
- Large-scale data communication

Step 3: Less Signal Loss
Optical fiber experiences very little signal loss even over long distances. In contrast, coaxial cables suffer from signal attenuation, which requires signal boosters or repeaters.
Thus, optical fiber can transmit data over much longer distances without degradation.
Step 4: Immunity to Electromagnetic Interference
Optical fibers are not affected by electromagnetic interference (EMI) because they use light rather than electrical signals.

- No interference from electrical equipment
- More reliable data transmission

Coaxial cables, on the other hand, can be affected by electromagnetic noise.
Step 5: Better Security
Optical fiber is more secure for data transmission. It is difficult to tap into fiber cables without being detected, which makes them ideal for secure communication networks.
Step 6: Summary of Advantages

Feature	Optical Fiber	Coaxial Cable
Signal Type	Light signals	Electrical signals
Speed	Very high	Moderate
Bandwidth	Very high	Lower
Signal Loss	Very low	Higher
Interference	Immune to EMI	Affected by EMI
Security	High	Moderate

Step 7: Conclusion
Optical fiber is superior to coaxial cable because it provides faster data transmission, higher bandwidth, better security, lower signal loss, and immunity to electromagnetic interference. These advantages make optical fiber the preferred medium for modern communication networks.

Quick Tip

Key point to remember:

- **Optical Fiber** → Uses light signals
- **Coaxial Cable** → Uses electrical signals

Optical fiber offers **higher speed, greater bandwidth, and better reliability**.

---

11. Explain Method Overloading in the context of polymorphism.

Solution:

Concept:
Polymorphism is one of the fundamental concepts of Object-Oriented Programming (OOP). The word polymorphism means **"many forms."** It allows the same function or method name to perform different tasks depending on the context.
One way to achieve polymorphism is through **Method Overloading**. Method overloading occurs when multiple methods have the **same name** but **different parameters** (different number or types of arguments). The correct method is selected automatically based on the arguments passed during the function call.
Thus, method overloading increases code readability and allows programmers to use a single method name for related operations.
Step 1: Understanding Method Overloading.
In method overloading:

- The method name remains the **same**.
- The **number or type of parameters** changes.
- The program determines which method to execute based on the arguments provided.

Example situations:

- Adding two numbers
- Adding three numbers
- Adding floating-point numbers

All these operations can use the same method name but different parameter lists.
Step 2: Example illustrating Method Overloading.
Although Python does not support traditional method overloading directly, similar behavior can be implemented using default arguments.

```
class Calculator:

    def add(self, a, b, c = 0):
        return a + b + c

obj = Calculator()

print(obj.add(2, 3))      # Adding two numbers
print(obj.add(2, 3, 4))   # Adding three numbers
```

Output:

```
5
9
```

In this example, the same method `add()` performs different operations depending on the number of arguments provided.

Step 3: Advantages of Method Overloading.

- Improves **code readability**.
- Reduces the need for multiple method names.
- Supports **polymorphism** by allowing the same interface to perform different tasks.

Step 4: Conclusion.

Method overloading is a feature of polymorphism where multiple methods share the same name but differ in parameters. It allows programmers to perform different operations using the same method name, making programs easier to understand and maintain.

Quick Tip

Key idea:

- **Polymorphism** → One interface, many forms
- **Method Overloading** → Same method name with different parameters

Example:

```
add(a, b)
add(a, b, c)
```

---

12. Write a Python program to find the factorial of a number using a recursive function.

Solution:

Concept:

The **factorial** of a number is the product of all positive integers less than or equal to that number. It is represented using the symbol !.

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 1$$

Example:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

A **recursive function** is a function that calls itself during its execution. Recursion is commonly used to solve problems that can be broken down into smaller subproblems of the same type.
Step 1: Understanding recursion in factorial calculation.
The factorial function can be defined recursively as:

$$n! = n \times (n-1)!$$

with the base condition:

$$0! = 1 \quad \text{and} \quad 1! = 1$$

The base condition stops the recursion.
Step 2: Python program using recursion.

```
# Recursive function to find factorial

def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Taking input from user
num = int(input("Enter a number: "))

# Calculating factorial
result = factorial(num)

print("Factorial of", num, "is", result)
```

Step 3: Example execution.
If the user enters:

```
Enter a number: 5
```

The output will be:

```
Factorial of 5 is 120
```

Step 4: How recursion works in this program.
For $n = 5$:

```
factorial(5)
= 5 * factorial(4)
= 5 * 4 * factorial(3)
= 5 * 4 * 3 * factorial(2)
= 5 * 4 * 3 * 2 * factorial(1)
= 5 * 4 * 3 * 2 * 1
= 120
```

Quick Tip

Important points about recursion:

- A recursive function calls itself.
- It must always have a **base condition**.
- Without a base condition, the program may run infinitely.

Example base condition for factorial:

$$0! = 1$$

---

13. Define Cyber Ethics and list two common netiquettes every user should follow.

Solution:

Concept:
Cyber Ethics refers to the moral principles and responsible behavior that individuals should follow while using computers, the internet, and digital technologies. It involves using online resources in a legal, safe, and respectful manner.
Cyber ethics guides users on how to behave properly in the digital world, ensuring that technology is used responsibly without harming others or violating laws. It includes respecting privacy, protecting intellectual property, and avoiding harmful activities such as hacking, cyberbullying, and spreading false information.
Step 1: Understanding Cyber Ethics.
Cyber ethics promotes responsible online behavior such as:

- Respecting other users on the internet.
- Avoiding illegal activities such as hacking or piracy.
- Protecting personal and sensitive information.
- Using digital resources in a responsible and ethical way.

Following cyber ethics helps maintain a safe and positive digital environment.
Step 2: Understanding Netiquette.

Netiquette (Network Etiquette) refers to the set of rules and guidelines for polite and respectful behavior while communicating online through emails, social media, forums, or messaging platforms.
These rules help maintain proper communication and prevent misunderstandings in online interactions.
Step 3: Two common netiquettes every user should follow.

- **Be respectful and polite:** Always communicate respectfully with others and avoid using offensive or abusive language in online conversations.
- **Respect privacy:** Do not share someone else's personal information, photos, or messages without their permission.

Step 4: Conclusion.
Cyber ethics encourages responsible and ethical use of digital technology, while netiquette provides guidelines for polite and respectful online communication. Following these principles helps create a safe and positive online environment.

Quick Tip

Important ideas to remember:

- **Cyber Ethics** → Moral rules for using computers and the internet.
- **Netiquette** → Proper behavior while communicating online.

Example:
Be respectful, avoid cyberbullying, and protect personal information online.