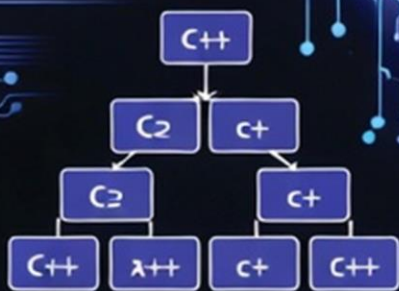


# NCERT Solutions for Class 12 COMPUTER SCIENCE



**Class 12**

**CBSE**



# **Computer Science – Class XII**

## **Detailed Exercise Solutions**

*NCERT Textbook | All Chapters*



<https://pagenest.in/>

# CONTENTS

# Chapter 1: Exception Handling in Python

*This chapter deals with errors and exceptions — situations where a program encounters something unexpected. Learning to handle exceptions makes your programs more robust and user-friendly.*

**Q1. "Every syntax error is an exception but every exception cannot be a syntax error." Justify the statement.**

**Answer:**

This statement is TRUE. Here is why:

**Syntax Error → Is it an Exception? YES.**

A syntax error means you have written code that Python cannot understand because it violates the grammar rules of the language (e.g., forgetting a colon, wrong indentation). Python raises a `SyntaxError` — which is itself a sub-class of `Exception`. So every syntax error IS a type of exception.

**Every Exception → Is it a Syntax Error? NO.**

Many exceptions arise during program execution (runtime) even when the code is grammatically correct. For example:

- `ZeroDivisionError`: When you try to divide by zero — the code is syntactically correct, but mathematically invalid.
- `NameError`: Using a variable before defining it.
- `FileNotFoundError`: Trying to open a file that does not exist.

*Think of it like this: All squares are rectangles, but not all rectangles are squares. Similarly, all syntax errors are exceptions, but not all exceptions are syntax errors.*

**Q2. When are the following built-in exceptions raised? Give examples.**

**Answer:**

**a) ImportError:**

Raised when Python cannot find or load a module you are trying to import.

```
|import nonexistent_module    # ImportError: No module named
|'nonexistent_module'
```

**b) IOError (OSError):**

Raised when an input/output operation fails, such as opening a file that does not exist.

```
|f = open('missing_file.txt', 'r')    # IOError/FileNotFoundError
```

**c) NameError:**

Raised when you use a variable or function name that has not been defined yet.

```
|print(undeclared_var)    # NameError: name 'undeclared_var' is not defined
```

**d) ZeroDivisionError:**

Raised when you try to divide any number by zero.

```
|result = 10 / 0    # ZeroDivisionError: division by zero
```

**Q3. What is the use of a raise statement? Write code to accept two numbers and display quotient, raising exception if denominator is zero.**

**Answer:**

The `raise` statement is used to explicitly trigger (force) an exception in your program. This is useful when you want to signal an error condition that Python would not automatically detect — for example, negative input where only positive numbers are valid.

```
try:
    num1 = int(input('Enter first number: '))
    num2 = int(input('Enter second number: '))
    if num2 == 0:
        raise ZeroDivisionError('Denominator cannot be zero!')
    quotient = num1 / num2
    print('Quotient =', quotient)
except ZeroDivisionError as e:
    print('Error:', e)
except ValueError:
    print('Please enter valid integers.')
```

The 'raise' keyword manually throws an exception. Even if Python does not raise it automatically, you can force it when your logic requires it.

#### Q4. Use assert statement in Question No. 3 to test the division expression.

##### Answer:

The assert statement checks a condition — if the condition is False, it raises an AssertionError. It is commonly used for testing/debugging.

```
try:
    num1 = int(input('Enter first number: '))
    num2 = int(input('Enter second number: '))
    assert num2 != 0, 'Denominator cannot be zero!'
    quotient = num1 / num2
    print('Quotient =', quotient)
except AssertionError as e:
    print('Assertion failed:', e)
except ValueError:
    print('Please enter valid integers.')
```

'assert condition, message' — if condition is True, nothing happens. If False, AssertionError is raised with the given message.

#### Q5. Define the following: a) Exception Handling b) Throwing an exception c) Catching an exception

##### Answer:

##### a) Exception Handling:

Exception handling is the mechanism in Python that allows a program to deal with unexpected runtime errors (exceptions) gracefully, instead of crashing. The program detects an error, handles it with appropriate code, and continues (or exits gracefully).

##### b) Throwing an Exception (Raising):

Throwing or raising an exception means deliberately generating an exception using the raise keyword or when Python automatically detects an error condition. Once thrown, if not handled, the program terminates.

##### c) Catching an Exception:

Catching means intercepting (handling) a raised exception using a try-except block. The except clause 'catches' the exception and executes alternative code rather than crashing.

#### Q6. Explain catching exceptions using try and except block.

##### Answer:

The try-except block is the primary mechanism for exception handling:

```
try:
    # Code that might raise an exception
```

```

    risky_code()
except ExceptionType:
    # Code to run if the above exception occurs
    handle_error()

```

How it works step-by-step:

1. Python executes the try block statement by statement.
2. If NO exception occurs, the except block is SKIPPED.
3. If an exception DOES occur, Python immediately jumps to the matching except block.
4. After the except block runs, execution continues normally after the try-except.

Multiple exceptions can be handled:

```

try:
    x = int(input('Enter number: '))
    y = 10 / x
except ValueError:
    print('Not a valid number!')
except ZeroDivisionError:
    print('Cannot divide by zero!')

```

**Q7. Consider the code given below and fill in the blanks.**

**Answer:**

```

print("Learning Exceptions...")
try:
    num1 = int(input('Enter the first number'))
    num2 = int(input('Enter the second number'))
    quotient = (num1 / num2)
    print("Both the numbers entered were correct")
except ValueError:          # to enter only integers
    print("Please enter only numbers")
except ZeroDivisionError:  # Denominator should not be zero
    print("Number 2 should not be zero")
else:
    print("Great .. you are a good programmer")
finally:                   # to be executed at the end
    print("JOB OVER... GO GET SOME REST")

```

*Fill-in answers: 'ValueError', 'ZeroDivisionError', 'finally:'*

**Q8. Write code to use wrong number of arguments for math.sqrt() and catch ValueError.**

**Answer:**

```

import math
try:
    result = math.sqrt(16, 2)    # sqrt() takes only 1 argument
    print(result)
except TypeError as e:
    print('TypeError caught:', e)
except ValueError as e:
    print('ValueError caught:', e)

```

*Note: Passing wrong number of args to sqrt() raises TypeError, not ValueError. ValueError is raised by math.sqrt(-1) since square root of negative is not real.*

```

import math
try:
    result = math.sqrt(-9)      # ValueError: math domain error
    print(result)

```

```
except ValueError as e:  
    print('ValueError caught:', e)
```

**Q9. What is the use of finally clause? Use finally clause in Question 7.**

**Answer:**

The finally clause contains code that ALWAYS executes — whether an exception occurred or not. It is typically used for cleanup operations like closing files, releasing resources, or printing final status.

*Refer to Question 7 solution above — it already includes the finally clause: finally: print('JOB OVER... GO GET SOME REST')*

## Chapter 2: File Handling in Python

File handling allows programs to store data permanently on disk and retrieve it later. This chapter covers reading, writing, and manipulating text and binary files.

### Q1. Differentiate between:

**Answer:**

#### a) Text File vs Binary File:

| Feature     | Text File                                          | Binary File                           |
|-------------|----------------------------------------------------|---------------------------------------|
| Storage     | Stores data as readable characters (ASCII/Unicode) | Stores data as raw bytes (0s and 1s)  |
| Readability | Human-readable (can open in Notepad)               | Not human-readable directly           |
| Extension   | .txt, .csv, .py                                    | .dat, .bin, .jpg, .mp3                |
| Line ending | Translated (\n becomes platform specific)          | No translation – exact bytes stored   |
| Use case    | Config files, logs, source code                    | Images, audio, pickled Python objects |

#### b) readline() vs readlines():

| Method  | readline()                   | readlines()                       |
|---------|------------------------------|-----------------------------------|
| Reads   | One line at a time           | All lines at once                 |
| Returns | A string (one line)          | A list of strings (all lines)     |
| Usage   | When processing line by line | When you need all lines in memory |

#### c) write() vs writelines():

| Method   | write()                       | writelines()                     |
|----------|-------------------------------|----------------------------------|
| Accepts  | A single string               | A list of strings                |
| Newlines | Does NOT add \n automatically | Does NOT add \n automatically    |
| Example  | f.write('Hello')              | f.writelines(['Hello', 'World']) |

### Q2. Write the use and syntax for the following methods: a) open() b) read() c) seek() d) dump()

**Answer:**

#### a) open():

Use: Opens a file and returns a file object. Required before any file operation.

```
file_object = open(filename, mode)
```

Modes: 'r' (read), 'w' (write), 'a' (append), 'rb' (read binary), 'wb' (write binary)

**b) read():**

Use: Reads the entire content of a text file as a single string.

```
content = file_object.read()      # reads entire file
content = file_object.read(n)     # reads n characters
```

**c) seek():**

Use: Moves the file pointer to a specific position. Useful for re-reading or jumping to a location.

```
file_object.seek(0)               # Go to beginning of file
file_object.seek(10)              # Go to 10th byte
```

**d) dump():**

Use: From the pickle module. Serializes (converts) a Python object to bytes and writes it to a binary file.

```
import pickle
pickle.dump(object, binary_file_object)
```

**Q3. Write the file mode for opening the following files and the Python statements.****Answer:**

```
# a) text file 'example.txt' in both read and write mode
f = open('example.txt', 'r+')   # 'r+' opens for both reading and writing

# b) binary file 'bfile.dat' in write mode
f = open('bfile.dat', 'wb')     # 'wb' = write binary

# c) text file 'try.txt' in append and read mode
f = open('try.txt', 'a+')       # 'a+' = append + read

# d) binary file 'btry.dat' in read only mode
f = open('btry.dat', 'rb')      # 'rb' = read binary
```

**Q4. Why is it advised to close a file after we are done? What happens if we do not close it?****Answer:**

It is advised to close a file after use for the following reasons:

1. Data Integrity: Python may buffer (store temporarily in memory) data before writing to the file. If you don't close, buffered data may NOT be written to disk — leading to data loss.
2. Resource Release: Each open file uses system resources (file handles). Not closing files eventually leads to running out of available file handles.
3. Locking: Some operating systems lock files while they are open, preventing other programs from accessing them.

If you do NOT close:

- No immediate error message is usually shown.
- Data might not be saved properly.
- When the program ends, Python's garbage collector eventually closes the file — but this is unreliable.

*Best practice: Use the 'with' statement which automatically closes the file even if an error occurs.*

**Q5. What is the difference between the two sets of statements (a) and (b)?****Answer:**

```
# (a) Traditional open
P = open('practice.txt', 'r')
P.read(10)    # Reads only 10 characters
# NOTE: File is NOT automatically closed after this block!
```

```
# (b) Using 'with' statement
with open('practice.txt', 'r') as P:
    x = P.read()    # Reads the ENTIRE file
```

Differences:

| Aspect       | Statement (a)                | Statement (b)                           |
|--------------|------------------------------|-----------------------------------------|
| Reading      | Reads only 10 characters     | Reads the entire file                   |
| File closing | Must call P.close() manually | Automatically closed after 'with' block |
| Safety       | Risk of not closing file     | File always closed, even on error       |
| Style        | Older approach               | Modern, recommended approach            |

**Q6. Write commands to write the following lines to hello.txt (file opened in append mode).**

**Answer:**

```
# Method 1: Using writelines()
with open('hello.txt', 'a') as f:
    f.writelines(["Welcome my class\n", "It is a fun place\n", "You will learn and play\n"])

# Method 2: Using write() for each line
with open('hello.txt', 'a') as f:
    f.write('Welcome my class\n')
    f.write('It is a fun place\n')
    f.write('You will learn and play\n')
```

*'a' mode = append. New content is added to the END of existing file without deleting previous content.*

**Q7. Write a Python program to open hello.txt in read mode and display its contents. What difference if opened in write mode?**

**Answer:**

```
# Reading the file
with open('hello.txt', 'r') as f:
    content = f.read()
    print(content)
```

If opened in write mode ('w') instead of append mode ('a'):

- 'a' (append): Content is added to the END of the file. Existing content is PRESERVED.
- 'w' (write): The file is CLEARED first, then new content is written. All previous content is LOST.

**Q8. Write a program to accept sentences until user types 'END'. Save to text file and display sentences beginning with uppercase.**

**Answer:**

```
# Writing sentences to file
with open('sentences.txt', 'w') as f:
    while True:
        line = input('Enter a sentence (or END to stop): ')
        if line.upper() == 'END':
            break
        f.write(line + '\n')
```

```
# Reading and displaying sentences starting with uppercase
print('\nSentences starting with uppercase:')
with open('sentences.txt', 'r') as f:
    for sentence in f:
        sentence = sentence.strip()
        if sentence and sentence[0].isupper():
            print(sentence)
```

### Q9. Define pickling in Python. Explain serialization and deserialization.

**Answer:**

#### Pickling:

Pickling is the process of converting a Python object (list, dictionary, class instance, etc.) into a byte stream so it can be stored in a binary file. The pickle module handles this.

#### Serialization:

Serialization = Converting a Python object—byte stream (for storage/transmission). Done using pickle.dump().

#### Deserialization:

Deserialization = Converting a byte stream—back to Python object. Done using pickle.load().

```
import pickle

# Serialization - writing object to binary file
data = {'name': 'Alice', 'age': 17, 'marks': [95, 88, 92]}
with open('data.pkl', 'wb') as f:
    pickle.dump(data, f)
    print('Object serialized and saved.')

# Deserialization - reading object from binary file
with open('data.pkl', 'rb') as f:
    retrieved = pickle.load(f)
    print('Deserialized object:', retrieved)
```

### Q10. Write a program to enter records in a binary file and display them with Amount calculated.

**Answer:**

```
import pickle

n = int(input('How many records to enter? '))
records = []

for i in range(n):
    print(f'\nRecord {i+1}:')
    item_no = int(input('  Item No: '))
    item_name = input('  Item Name: ')
    qty = int(input('  Quantity: '))
    price = float(input('  Price: '))
    records.append({'ItemNo': item_no, 'ItemName': item_name,
                    'Qty': qty, 'Price': price})

# Write to binary file
with open('inventory.dat', 'wb') as f:
    pickle.dump(records, f)
    print('\nRecords saved successfully!')
```

```
# Read and display records
with open('inventory.dat', 'rb') as f:
    data = pickle.load(f)

print('\n--- Inventory Records ---')
for rec in data:
    amount = rec['Price'] * rec['Qty']
    print(f'Item No      : {rec["ItemNo"]}')
    print(f'Item Name     : {rec["ItemName"]}')
    print(f'Quantity        : {rec["Qty"]}')
    print(f'Price per item: {rec["Price"]}')
    print(f'Amount          : {amount}')
    print('-' * 30)
```

## Chapter 3: Stack

*A Stack is a linear data structure that follows the LIFO (Last In, First Out) principle — the last element inserted is the first to be removed, like a stack of plates.*

### Q1. State TRUE or FALSE:

**Answer:**

- a) Stack is a linear data structure—TRUE  
(Stack stores elements in a linear/sequential manner)
- b) Stack does not follow LIFO rule—FALSE  
(Stack specifically follows LIFO — Last In, First Out)
- c) PUSH operation may result into underflow condition—FALSE  
(PUSH adds an element. OVERFLOW occurs if stack is full. UNDERFLOW occurs during POP when stack is empty)
- d) In POSTFIX notation, operators are placed after operands—TRUE  
(e.g., A + B in infix becomes A B + in postfix)

### Q2. Find the output of the following code:

**Answer:**

a)

```
result = 0
numberList = [10, 20, 30]
numberList.append(40)      # List: [10, 20, 30, 40]
result = result + numberList.pop() # pops 40 → result = 40
result = result + numberList.pop() # pops 30 → result = 70
print("Result=", result)    # Output: Result= 70
```

**Output: Result= 70**

b)

```
answer = []; output = ''
answer.append('T') # ['T']
answer.append('A') # ['T', 'A']
answer.append('M') # ['T', 'A', 'M']
ch = answer.pop() # ch = 'M', answer = ['T', 'A']
output = output + ch # output = 'M'
ch = answer.pop() # ch = 'A', answer = ['T']
output = output + ch # output = 'MA'
ch = answer.pop() # ch = 'T', answer = []
output = output + ch # output = 'MAT'
print("Result=", output) # Output: Result= MAT
```

**Output: Result= MAT**

*Notice: The stack reverses the order — 'T','A','M' pushed, 'M','A','T' popped → 'MAT'*

### Q3. Write a program to reverse a string using stack.

**Answer:**

*Concept: Push each character onto a stack. Then pop them all — since stack is LIFO, the string comes out reversed.*

```
def reverse_string(s):
    stack = []
```

```

# Push all characters
for char in s:
    stack.append(char)
# Pop all characters to get reversed string
reversed_str = ''
while stack:
    reversed_str += stack.pop()
return reversed_str

# Main program
text = input('Enter a string: ')
result = reverse_string(text)
print('Reversed string:', result)

```

**Q4. For expression  $((2+3)*(4/2))+2$  — show step-by-step parenthesis matching using stack.**

**Answer:**

Algorithm: Push '(' onto stack. When ')' is found, pop from stack. If mismatch or empty stack, parentheses are unbalanced.

| Step  | Character | Stack State | Action                                      |
|-------|-----------|-------------|---------------------------------------------|
| 1     | (         | [ ( ]       | Push '('                                    |
| 2     | (         | [ (, ( ]    | Push '('                                    |
| 3     | 2         | [ (, ( ]    | Ignore (not bracket)                        |
| 4     | +         | [ (, ( ]    | Ignore                                      |
| 5     | 3         | [ (, ( ]    | Ignore                                      |
| 6     | )         | [ ( ]       | Pop '(' — inner pair matched ✓              |
| 7     | *         | [ ( ]       | Ignore                                      |
| 8     | (         | [ (, ( ]    | Push '('                                    |
| 9     | 4         | [ (, ( ]    | Ignore                                      |
| 10    | /         | [ (, ( ]    | Ignore                                      |
| 11    | 2         | [ (, ( ]    | Ignore                                      |
| 12    | )         | [ ( ]       | Pop '(' — inner pair matched ✓              |
| 13    | )         | [ ]         | Pop '(' — outer pair matched ✓              |
| 14    | +         | [ ]         | Ignore                                      |
| 15    | 2         | [ ]         | Ignore                                      |
| Final | -         | [ ] (empty) | Stack is empty → All parentheses balanced ✓ |

[Diagram prompt: Create a diagram showing a stack represented as a vertical box, with arrows showing push/pop operations at each step for the above expression]

**Q5. Evaluate postfix expressions (A=3, B=5, C=1, D=4):**

**Answer:**

**a)  $A B + C *$  (i.e.,  $(A+B)*C$ )**

| Token  | Action                                | Stack After |
|--------|---------------------------------------|-------------|
| A (=3) | Push 3                                | [3]         |
| B (=5) | Push 5                                | [3, 5]      |
| +      | Pop 5 and 3, compute $3+5=8$ , push 8 | [8]         |
| C (=1) | Push 1                                | [8, 1]      |
| *      | Pop 1 and 8, compute $8*1=8$ , push 8 | [8]         |
| End    | Result = pop() = 8                    | []          |

**Result = 8****b)  $A B * C / D *$  (i.e.,  $((A*B)/C)*D$ )**

| Token  | Action                                    | Stack After |
|--------|-------------------------------------------|-------------|
| A (=3) | Push 3                                    | [3]         |
| B (=5) | Push 5                                    | [3, 5]      |
| *      | Pop 5 and 3, compute $3*5=15$ , push 15   | [15]        |
| C (=1) | Push 1                                    | [15, 1]     |
| /      | Pop 1 and 15, compute $15/1=15$ , push 15 | [15]        |
| D (=4) | Push 4                                    | [15, 4]     |
| *      | Pop 4 and 15, compute $15*4=60$ , push 60 | [60]        |
| End    | Result = pop() = 60                       | []          |

**Result = 60****Q6. Convert infix to postfix (showing stack and string contents at each step):****Answer:****a)  $A + B - C * D$** 

Operator precedence: \* and / &gt; + and -

| Step | Character | Stack | Postfix Output | Notes                                                |
|------|-----------|-------|----------------|------------------------------------------------------|
| 1    | A         | []    | A              | Operand → directly to output                         |
| 2    | +         | [+]   | A              | Push + (stack empty)                                 |
| 3    | B         | [+]   | AB             | Operand → output                                     |
| 4    | -         | [-]   | AB+            | - has same precedence as +, pop + first, then push - |

|     |   |        |         |                                        |
|-----|---|--------|---------|----------------------------------------|
| 5   | C | [-]    | AB+C    | Operand → output                       |
| 6   | * | [-, *] | AB+C    | * has higher precedence than -, push * |
| 7   | D | [-, *] | AB+CD   | Operand → output                       |
| End | - | []     | AB+CD*- | Pop remaining: *, then -               |

**Postfix Result: AB+CD\*-**

**b) A \* (( C + D ) / E)**

| Step | Character | Stack           | Postfix Output | Notes                |
|------|-----------|-----------------|----------------|----------------------|
| 1    | A         | []              | A              | Operand              |
| 2    | *         | [*]             | A              | Push *               |
| 3    | (         | [* , (]         | A              | Push opening bracket |
| 4    | (         | [* , ( , (]     | A              | Push opening bracket |
| 5    | C         | [* , ( , (]     | AC             | Operand              |
| 6    | +         | [* , ( , ( , +] | AC             | Push +               |
| 7    | D         | [* , ( , ( , +] | ACD            | Operand              |
| 8    | )         | [* , (]         | ACD+           | Pop until ( → pops + |
| 9    | /         | [* , ( , /]     | ACD+           | Push /               |
| 10   | E         | [* , ( , /]     | ACD+E          | Operand              |
| 11   | )         | [*]             | ACD+E/         | Pop until ( → pops / |
| End  | -         | []              | ACD+E/*        | Pop remaining: *     |

**Postfix Result: ACD+E/\***

**Q7. Write a program to create a Stack storing only odd numbers. Display contents and largest odd number.**

**Answer:**

```
def push(stack, item):
    stack.append(item)

def pop(stack):
    if not stack:
        return None
    return stack.pop()

def display(stack):
    print('Stack contents (top to bottom):')
```

```

        for i in range(len(stack)-1, -1, -1):
            print(stack[i])

# Main program
stack = []
n = int(input('How many numbers? '))

for i in range(n):
    num = int(input(f'Enter number {i+1}: '))
    if num % 2 != 0:    # Only push odd numbers
        push(stack, num)

display(stack)

# Find largest by popping
largest = None
temp = stack.copy()
while temp:
    item = pop(temp)
    if largest is None or item > largest:
        largest = item

if largest is not None:
    print('Largest odd number:', largest)
else:
    print('No odd numbers were entered.')

```

## Chapter 4: Queue

*A Queue is a linear data structure that follows FIFO (First In, First Out) principle — the first element inserted is the first to be removed, like a queue of people at a bank.*

### Q1. Fill in the blanks:

#### Answer:

- QUEUE is a linear list of elements in which insertion and deletion takes place from different ends.
- Operations on a queue are performed in FIFO (First In, First Out) order.
- Insertion operation in a queue is called ENQUEUE and deletion operation is called DEQUEUE.
- Deletion of elements is performed from the FRONT end of the queue.
- Elements 'A','S','D','F' are deleted one at a time—sequence received: A, S, D, F (FIFO order).
- DEQUE (Double Ended Queue) is a data structure where elements can be added or removed at either end.
- Deque contains 'z','x','c','v','b'. Deletions received: 'z','b','v','x','c'.

Sequence of operations: deletionFront, deletionRear, deletionRear, deletionRear, deletionFront

### Q2. Compare and contrast queue with stack.

#### Answer:

| Feature       | Stack                                  | Queue                              |
|---------------|----------------------------------------|------------------------------------|
| Principle     | LIFO - Last In, First Out              | FIFO - First In, First Out         |
| Insertion end | TOP only                               | REAR end                           |
| Deletion end  | TOP only                               | FRONT end                          |
| Operations    | PUSH (insert), POP (delete)            | ENQUEUE (insert), DEQUEUE (delete) |
| Real life     | Pile of plates, back button in browser | Queue at bank, print spooler       |
| Peek          | Views TOP element                      | Views FRONT element                |

### Q3. How does FIFO describe queue?

#### Answer:

FIFO stands for First In, First Out. In a queue, the element that enters FIRST is the one that gets removed FIRST. This is exactly how real-world queues work:

- At a bank: the customer who arrives first is served first.
- In a print queue: the first document sent to the printer gets printed first.

*[Diagram: See Figure 4.1 in textbook showing queue of people at a bank — the person at the FRONT is served next, and new people join at the REAR]*

### Q4. Write a menu driven python program using queue to implement movement of shuttlecock in its box.

#### Answer:

*Concept: Shuttlecocks are placed in a cylindrical box — you take from the top (FRONT) and add to the top. However, in a standard queue model, we treat adding at REAR and removing from FRONT.*

```
from collections import deque
```

```

def menu():
    shuttleBox = deque()
    while True:
        print('\n--- Shuttlecock Box ---')
        print('1. Add shuttlecock')
        print('2. Remove shuttlecock')
        print('3. Display box')
        print('4. Exit')
        choice = int(input('Enter choice: '))
        if choice == 1:
            name = input('Enter shuttlecock name/number: ')
            shuttleBox.append(name)
            print(f'{name} added to box.')
        elif choice == 2:
            if shuttleBox:
                removed = shuttleBox.popleft()
                print(f'{removed} removed from box.')
            else:
                print('Box is empty!')
        elif choice == 3:
            if shuttleBox:
                print('Shuttlecocks in box:', list(shuttleBox))
            else:
                print('Box is empty.')
        elif choice == 4:
            print('Exiting...')
            break
        else:
            print('Invalid choice!')

menu()

```

### Q5. How is queue data type different from deque data type?

**Answer:**

| Feature      | Queue                 | Deque (Double-Ended Queue)    |
|--------------|-----------------------|-------------------------------|
| Insertion    | Only at REAR          | At BOTH ends (front and rear) |
| Deletion     | Only at FRONT         | At BOTH ends (front and rear) |
| Flexibility  | Less flexible         | More flexible                 |
| Can act as   | Queue only            | Both queue AND stack          |
| Python class | Custom or queue.Queue | collections.deque             |

### Q6. Show the status of queue after each operation:

**Answer:**

| Operation   | Queue State (front → rear) | Notes            |
|-------------|----------------------------|------------------|
| Start       | [] (empty)                 |                  |
| enqueue(34) | [34]                       | 34 at rear/front |
| enqueue(54) | [34, 54]                   | 54 added at rear |

|             |                 |                                |
|-------------|-----------------|--------------------------------|
| dequeue()   | [54]            | 34 removed from front          |
| enqueue(12) | [54, 12]        | 12 added at rear               |
| dequeue()   | [12]            | 54 removed from front          |
| enqueue(61) | [12, 61]        | 61 added at rear               |
| peek()      | [12, 61]        | Returns 12 (front), no removal |
| dequeue()   | [61]            | 12 removed                     |
| dequeue()   | []              | 61 removed                     |
| dequeue()   | [] (Underflow!) | Queue already empty            |
| dequeue()   | [] (Underflow!) | Queue already empty            |
| enqueue(1)  | [1]             | 1 added                        |

**Q7. Show the status of deque after each operation:**

**Answer:**

| Operation       | Deque State     | Notes                           |
|-----------------|-----------------|---------------------------------|
| Start           | [] (empty)      |                                 |
| peek()          | [] → Error/None | Deque is empty, nothing to peek |
| insertFront(12) | [12]            | 12 inserted at front            |
| insertRear(67)  | [12, 67]        | 67 inserted at rear             |
| deletionFront() | [67]            | 12 removed from front           |
| insertRear(43)  | [67, 43]        | 43 inserted at rear             |
| deletionRear()  | [67]            | 43 removed from rear            |
| deletionFront() | []              | 67 removed from front           |
| deletionRear()  | [] → Error/None | Deque is empty                  |

**Q8. Write a Python program to check whether given string is palindrome or not using deque.**

**Answer:**

*Concept: Add all characters to a deque. Then repeatedly remove from both front and rear and compare. If all pairs match and deque has 0 or 1 element at end, it's a palindrome.*

```
from collections import deque

def is_palindrome(s):
    # Remove spaces and convert to lowercase
    s = s.replace(' ', '').lower()
```

```

dq = deque(s)

while len(dq) > 1:
    front = dq.popleft()    # Remove from front
    rear  = dq.pop()        # Remove from rear
    if front != rear:
        return False
    return True

# Main program
word = input('Enter a string: ')
if is_palindrome(word):
    print(f'"{word}" is a Palindrome!')
else:
    print(f'"{word}" is NOT a Palindrome.')

# Test: 'racecar' → Palindrome, 'hello' → Not a Palindrome

```

## Chapter 5: Sorting

Sorting is the process of arranging elements in a specific order (ascending or descending). This chapter covers Bubble Sort, Selection Sort, and Insertion Sort.

**Q1. Display partially sorted list after 3 complete passes of Bubble Sort for: numList = [7,11,3,10,17,23,1,4,21,5]**

**Answer:**

Bubble Sort: In each pass, adjacent elements are compared and swapped if out of order. The largest unsorted element 'bubbles up' to its correct position after each pass.

**Original List: [7, 11, 3, 10, 17, 23, 1, 4, 21, 5]**

**Pass 1:**

Compare adjacent pairs from left to right:

7,11—OK | 11,3—Swap | 11,10—Swap | 11,17—OK | 17,23—OK | 23,1—Swap | 23,4—Swap |  
23,21—Swap | 23,5—Swap

**After Pass 1: [7, 3, 10, 11, 17, 1, 4, 21, 5, 23]**

*23 (largest) is now at the correct end position.*

**Pass 2:**

7,3—Swap | 7,10—OK | 10,11—OK | 11,17—OK | 17,1—Swap | 17,4—Swap | 17,21—OK | 21,5—Swap

**After Pass 2: [3, 7, 10, 11, 1, 4, 17, 5, 21, 23]**

*21 is now also in correct position.*

**Pass 3:**

3,7—OK | 7,10—OK | 10,11—OK | 11,1—Swap | 11,4—Swap | 11,17—OK | 17,5—Swap

**After Pass 3: [3, 7, 10, 1, 4, 11, 5, 17, 21, 23]**

**Q2. Identify number of swaps for List: 63 42 21 9 using Selection Sort and Bubble Sort.**

**Answer:**

**Bubble Sort: [63, 42, 21, 9]**

Pass 1: ~~63,42~~, ~~63,21~~, ~~63,9~~ 3 swaps—[42, 21, 9, 63]

Pass 2: ~~42,21~~, ~~42,9~~ 2 swaps—[21, 9, 42, 63]

Pass 3: ~~21,9~~ 1 swap—[9, 21, 42, 63]

**Total Bubble Sort swaps: 6**

**Selection Sort: [63, 42, 21, 9]**

Pass 1: Find min (9), swap ~~63,9~~ 1 swap—[9, 42, 21, 63]

Pass 2: Find min (21), swap ~~42,21~~ 1 swap—[9, 21, 42, 63]

Pass 3: Find min (42), already in place—0 swaps—[9, 21, 42, 63]

**Total Selection Sort swaps: 2**

Conclusion: Selection Sort requires FEWER swaps (2 vs 6) for this list. Selection Sort is better when minimizing swaps is important.

**Q3. Compare Insertion Sort for List1: [2,3,5,7,11] vs List2: [11,7,5,3,2] — which makes minimum comparisons?**

**Answer:**

**List1: [2, 3, 5, 7, 11] — Already sorted (ascending)**

In each pass, the element is already in the right position — no shifting needed.

Pass 1: 3 compared with 2—1 comparison, no shift

Pass 2: 5 compared with 3—1 comparison, no shift

Pass 3: 7 compared with 5—1 comparison, no shift

Pass 4: 11 compared with 7—1 comparison, no shift

**Total comparisons for List1: 4 (minimum — best case for insertion sort)**

**List2: [11, 7, 5, 3, 2] — Sorted in reverse (worst case)**

Each element must be compared with ALL previously sorted elements.

Pass 1: 7 compared—1 comparison

Pass 2: 5 compared—2 comparisons

Pass 3: 3 compared—3 comparisons

Pass 4: 2 compared—4 comparisons

**Total comparisons for List2: 10 (maximum — worst case)**

**Conclusion: List1 makes MINIMUM comparisons (4) since it is already sorted.**

*[Diagram prompt: Draw two diagrams side-by-side showing insertion of each element step by step for both lists, with arrows indicating comparisons]*

**Q4. Write a program using user-defined functions to find median of a list using Bubble Sort.**

**Answer:**

```
def bubble_sort(lst):
    n = len(lst)
    for i in range(n-1):
        for j in range(n-1-i):
            if lst[j] > lst[j+1]:
                lst[j], lst[j+1] = lst[j+1], lst[j]
    return lst

def find_median(lst):
    sorted_lst = bubble_sort(lst.copy())
    n = len(sorted_lst)
    if n % 2 == 1: # Odd number of elements
        median = sorted_lst[n // 2]
    else: # Even number of elements
        median = (sorted_lst[n//2 - 1] + sorted_lst[n//2]) / 2
    return median

# Main
n = int(input('How many numbers? '))
numbers = []
for i in range(n):
    numbers.append(float(input(f'Enter number {i+1}: ')))

print('Original list:', numbers)
median = find_median(numbers)
print('Median:', median)
```

**Q5. Calculate xth percentile for n students using Selection Sort.**

**Answer:**

```

import math

def selection_sort(lst):
    n = len(lst)
    for i in range(n-1):
        min_idx = i
        for j in range(i+1, n):
            if lst[j] < lst[min_idx]:
                min_idx = j
        lst[i], lst[min_idx] = lst[min_idx], lst[i]
    return lst

def percentile(marks, x):
    sorted_marks = selection_sort(marks.copy())
    n = len(sorted_marks)
    index = math.ceil((x / 100) * n) # Calculate index
    index = min(index, n-1)         # Ensure within bounds
    return sorted_marks[index]

# Main
n = int(input('Number of students: '))
marks = [float(input(f'Enter marks of student {i+1}: ')) for i in range(n)]
x = float(input('Enter percentile value (0-100): '))

result = percentile(marks, x)
print(f'{x}th percentile mark = {result}')

```

**Q6. Students are inserted in ascending order during admission — identify sorting technique and write program.**

**Answer:**

This describes INSERTION SORT — each new element is placed at its correct sorted position among already-inserted elements.

```

def insert_in_order(sorted_list, new_name):
    # Find the correct position using insertion sort logic
    i = len(sorted_list) - 1
    sorted_list.append(new_name)
    while i >= 0 and sorted_list[i] > new_name:
        sorted_list[i+1] = sorted_list[i]
        i -= 1
    sorted_list[i+1] = new_name
    return sorted_list

# Main
student_list = []
n = int(input('How many students to admit? '))
for i in range(n):
    name = input(f'Enter student name {i+1}: ')
    student_list = insert_in_order(student_list, name)
    print('Current list:', student_list)

print('\nFinal sorted admission list:')
for i, name in enumerate(student_list, 1):
    print(f'{i}. {name}')

```

## Chapter 6: Searching

*Searching is the process of finding an element in a collection. This chapter covers Linear Search, Binary Search, and Hashing-based search.*

**Q1. Using linear search, find position of 8, 1, 99, 44 in list: [1,-2,32,8,17,19,42,13,0,44]**

**Answer:**

List: [1, -2, 32, 8, 17, 19, 42, 13, 0, 44] (Indices: 0–9)

**Searching for 8:**

| Pass | Index | List[index] | Compare with 8 | Decision                            |
|------|-------|-------------|----------------|-------------------------------------|
| 1    | 0     | 1           | 1 == 8?        | No                                  |
| 2    | 1     | -2          | -2 == 8?       | No                                  |
| 3    | 2     | 32          | 32 == 8?       | No                                  |
| 4    | 3     | 8           | 8 == 8?        | YES – Found at position 4 (index 3) |

**8 found at position 4 (index 3)**

**Searching for 1:**

Pass 1: index=0, list[0]=1, 1==1? YES—Found at position 1 (index 0)

**Searching for 99:**

Compare all 10 elements (1,-2,32,8,17,19,42,13,0,44)—99 not found.

**99 is NOT present in the list (10 comparisons made)**

**Searching for 44:**

Pass 1-9: 1,-2,32,8,17,19,42,13,0 not equal to 44

Pass 10: index=9, list[9]=44, 44==44? YES—Found at position 10 (index 9)

**44 found at position 10 (index 9)**

**Q2. Use linear search for key=8 in list with duplicate values: [42,-2,32,8,17,19,42,13,8,44]. What position is returned?**

**Answer:**

When a linear search finds a match, it typically returns the FIRST occurrence and stops.

```
def linear_search(lst, key):
    for i in range(len(lst)):
        if lst[i] == key:
            return i + 1 # Return 1-based position
    return -1

lst = [42, -2, 32, 8, 17, 19, 42, 13, 8, 44]
pos = linear_search(lst, 8)
print(f'Key 8 found at position: {pos}') # Returns 4
```

The position returned is 4 (index 3), which is the FIRST occurrence of 8. The second occurrence at index 8 is ignored because the search stops at the first match.

*This means that linear search returns the position of the first occurrence of a key when duplicates exist.*

**Q3. Write a program that takes a list of 10 negative/positive numbers and a key, applies linear search.**

**Answer:**

```
def linear_search(lst, key):
    for i in range(len(lst)):
        if lst[i] == key:
            return i + 1
    return -1

numbers = []
print('Enter 10 integers (can be negative or positive):')
for i in range(10):
    numbers.append(int(input(f'Enter number {i+1}: ')))

print('List:', numbers)
key = int(input('Enter key to search: '))

result = linear_search(numbers, key)
if result != -1:
    print(f'Key {key} found at position {result}')
else:
    print(f'Key {key} not found in the list.')
```

**Q4. Write a program that applies binary search on a list of 10 integers.**

**Answer:**

*Important: Binary search requires the list to be SORTED first.*

```
def binary_search(lst, key):
    low = 0
    high = len(lst) - 1
    while low <= high:
        mid = (low + high) // 2
        if lst[mid] == key:
            return mid + 1 # 1-based position
        elif lst[mid] < key:
            low = mid + 1
        else:
            high = mid - 1
    return -1

numbers = []
print('Enter 10 integers in SORTED order (ascending):')
for i in range(10):
    numbers.append(int(input(f'Enter number {i+1}: ')))

print('Sorted list:', numbers)
key = int(input('Enter key to search: '))

result = binary_search(numbers, key)
if result != -1:
    print(f'Key {key} found at position {result}')
else:
    print(f'Key {key} not found.')
```

**Q5. For the unsorted list [50,31,21,...99,44,55,9] — apply linear and binary search, then compare.**

**Answer:**

```
def linear_search_count(lst, key):
    for i in range(len(lst)):
        if lst[i] == key:
            return i+1, i+1 # position, comparisons
    return -1, len(lst)

def binary_search_count(lst, key):
    low, high, count = 0, len(lst)-1, 0
    while low <= high:
        count += 1
        mid = (low + high) // 2
        if lst[mid] == key: return mid+1, count
        elif lst[mid] < key: low = mid + 1
        else: high = mid - 1
    return -1, count

lst = [50,31,21,28,72,41,73,93,68,43,45,78,5,17,97,71,69,61,88,75,99,44,55,9]
keys = [1, 5, 55, 99]

print('--- Linear Search (Unsorted) ---')
for k in keys:
    pos, comps = linear_search_count(lst, k)
    print(f'Key {k}: Position={pos}, Comparisons={comps}')

sorted_lst = sorted(lst)
print('\n--- Linear Search (Sorted) ---')
for k in keys:
    pos, comps = linear_search_count(sorted_lst, k)
    print(f'Key {k}: Position={pos}, Comparisons={comps}')

print('\n--- Binary Search (Sorted) ---')
for k in keys:
    pos, comps = binary_search_count(sorted_lst, k)
    print(f'Key {k}: Position={pos}, Iterations={comps}')
```

**Q7. Estimate comparisons for binary vs linear search in a database of  $2^{30}$  records (key at middle).**

**Answer:**

**Database size:  $n = 2^{30} = 1,073,741,824$  records**

**Linear Search:**

In worst case (or when key is at middle), linear search scans records one by one.

For key at middle position: approximately  $n/2 = 2^{29} \approx 536,870,912$  comparisons.

**Binary Search:**

Binary search halves the list each time. Maximum comparisons =  $\log_2(n) = \log_2(2^{30}) = 30$  comparisons!

Interpretation: Binary search needs only 30 comparisons to search over a BILLION records, while linear search may need over 500 million. This shows why binary search is dramatically more efficient for large sorted datasets.

**Q8. Use hash function  $h(\text{element}) = \text{element} \% 11$  to store [44,121,55,33,110,77,22,66]. Search 11,44,88,121.**

**Answer:**

```
def hash_func(element, size=11):
    return element % size

numbers = [44, 121, 55, 33, 110, 77, 22, 66]
hash_table = [None] * 11

for num in numbers:
    index = hash_func(num)
    hash_table[index] = num
    print(f'{num} → index {index}')

print('\nHash Table:')
for i, val in enumerate(hash_table):
    print(f'Index {i}: {val}')

search_keys = [11, 44, 88, 121]
print('\nSearch Results:')
for key in search_keys:
    idx = hash_func(key)
    if hash_table[idx] == key:
        print(f'{key}: FOUND at index {idx}')
    else:
        print(f'{key}: NOT FOUND')
```

Hash table after insertion ( $h(x)=x\%11$ ):

| Index | Value                                   | Calculation                                                |
|-------|-----------------------------------------|------------------------------------------------------------|
| 0     | 22 or 44 or 33<br>or 110 or 77<br>or 66 | $44\%11=0$ , $33\%11=0$<br>(collision!), $110\%11=0\ldots$ |
| 0     | 44                                      | $44\%11=0$                                                 |
| 1     | 121                                     | $121\%11=0 \rightarrow$ Note: $121\%11=0$ ,<br>collision   |
| 5     | 55                                      | $55\%11=5$                                                 |
| 0     | 33                                      | $33\%11=0 \rightarrow$ collision                           |
| 0     | 110                                     | $110\%11=0 \rightarrow$ collision                          |
| 0     | 77                                      | $77\%11=0 \rightarrow$ collision                           |
| 0     | 22                                      | $22\%11=0 \rightarrow$ collision                           |
| 0     | 66                                      | $66\%11=0 \rightarrow$ collision                           |

*Note: Many elements map to index 0 (collision).  $121\%11 = 0$  as well. In simple hash table without collision resolution, only the last inserted value remains at that index.*

**Q9. Write a Python program for country-capital hash table using country name length as hash function.**

**Answer:**

```
CountryCapital = {
    'India': 'New Delhi',
    'UK': 'London',
```

```

    'France': 'Paris',
    'Switzerland': 'Berne',
    'Australia': 'Canberra'
}

# Create hash table (size = 12 to accommodate Switzerland)
size = 12
keys_list = [None] * size
values_list = [None] * size

# Hash function: length of country name
def hash_func(country):
    return len(country) - 1 # 0-indexed

# Insert into hash table
for country, capital in CountryCapital.items():
    idx = hash_func(country)
    keys_list[idx] = country
    values_list[idx] = capital

print('Hash Table:')
print(f'{"Index":<6} {"Country":<15} {"Capital":<15}')
for i in range(size):
    print(f'{i:<6} {str(keys_list[i]):<15} {str(values_list[i]):<15}')

# Search for capitals
search_countries = ['India', 'France', 'USA']
print('\nSearch Results:')
for country in search_countries:
    idx = hash_func(country)
    if idx < size and keys_list[idx] == country:
        print(f'Capital of {country}: {values_list[idx]}')
    else:
        print(f'{country}: NOT FOUND in hash table')

```

## Chapter 7: Understanding Data

*This chapter explores how data is collected, stored, processed, and analysed using statistical techniques like mean, median, mode, range, and standard deviation.*

### Q1. Identify data required for the following services:

**Answer:**

#### a) Declare exam results and print e-certificates:

Student ID/Roll Number, Name, Father's/Mother's Name, Subject marks, Total marks, Percentage, Grade/Division, School name, Exam date, Digital signature/seal

#### b) Register participants in an exhibition and issue biometric ID cards:

Name, Organization/Institution, Contact number, Email, Photo, Fingerprint/biometric data, Purpose of visit, Participant category, Registration date

#### c) To search for an image by a search engine:

Image metadata (size, format), Image features/visual embeddings, Tags/labels, Source URL, Upload date, Image hash for deduplication, Geo-tagging data (if any)

#### d) To book an OPD appointment with a hospital:

Patient name, Age, Gender, Contact number, Doctor name/Specialization, Department, Preferred date and time, Previous visit records, Insurance details (if any)

### Q2. A school wants to find beneficiaries of merit-cum-means scholarship. Describe data processing steps.

**Answer:**

Data required: Student ID, Name, Marks for Year 1, Marks for Year 2, Annual Family Income

Steps:

1. DATA COLLECTION: Gather academic records (marks sheets for 2 consecutive years) and family income certificates from all 500 students.
2. DATA INPUT & STORAGE: Enter data into a database/spreadsheet — columns: StudentID, Name, Year1\_Percentage, Year2\_Percentage, FamilyIncome.
3. DATA VALIDATION: Check for missing or incorrect entries (e.g., income > 0, marks between 0-100).
4. DATA PROCESSING/FILTERING: Apply conditions: Year1\_Percentage > 75 AND Year2\_Percentage > 75 AND FamilyIncome < 500000.
5. OUTPUT: Generate a list of students meeting all criteria — these are the beneficiaries.

### Q3. Bank wants to know its popularity based on number of accounts and average balance. Describe steps.

**Answer:**

1. Collect data: number of bank accounts per family in city 'ABC', average monthly balance per person.
2. Process: Calculate mean number of accounts per family, mean average balance.
3. Compare with competitor banks (if data available) to gauge relative popularity.
4. Summarize: Present findings in charts/graphs showing distribution of accounts and balances.

### Q4. Identify type of data being collected in the following scenarios:

**Answer:**

- a) Recording a video—Unstructured data (multimedia — audio + video combined)
- b) Marking attendance by teacher—Structured data (categorical: Present/Absent)

- c) Writing tweets—Semi-structured/Unstructured data (text with metadata like hashtags, timestamps)  
 d) Filling an application form online—Structured data (predefined fields: name, DOB, address, etc.)

**Q5. Temperature: 34,34,27,28,27,34,34. Identify technique for: a) average b) range c) standard deviation**

**Answer:**

Data: [34, 34, 27, 28, 27, 34, 34] (n = 7)

**a) Average Temperature → Use MEAN**

$$\text{Mean} = (34+34+27+28+27+34+34) / 7 = 218 / 7 \approx 31.14 \text{ }^{\circ}\text{C}$$

**b) Temperature Range → Use RANGE**

$$\text{Range} = \text{Maximum} - \text{Minimum} = 34 - 27 = 7 \text{ }^{\circ}\text{C}$$

**c) Standard Deviation → Use STANDARD DEVIATION**

Step 1: Mean = 31.14

Step 2: Squared deviations:  $(34-31.14)^2=8.18$ ,  $(34-31.14)^2=8.18$ ,  $(27-31.14)^2=17.14$ ,  $(28-31.14)^2=9.86$ ,  $(27-31.14)^2=17.14$ ,  $(34-31.14)^2=8.18$ ,  $(34-31.14)^2=8.18$

Step 3: Average of squared deviations =  $(8.18 \times 4 + 17.14 \times 2 + 9.86) / 7 = 76.72 / 7 \approx 10.96$

Step 4: SD =  $\sqrt{10.96} \approx 3.31 \text{ }^{\circ}\text{C}$

**Q6. A teacher wants to analyse results. Identify statistical technique for:**

**Answer:**

**a) Compare performance in terms of division secured in Class XII-A and XII-B:**

Use MODE — to find the most common division (e.g., First Division, Second Division) in each class and compare which class has more first-divisions.

**b) Compare class performance over 5 unit tests (July to November):**

Use MEAN — calculate average marks of the class in each unit test and compare across months to see trends.

**Q7. School annual day — felicitate parents who are alumni. Answer questions:**

**Answer:**

**a) Statistical technique to find students whose BOTH parents are alumni:**

Use MODE (frequency count) — count how many students have both/one/no parents as alumni. This is essentially a frequency/count-based technique.

**b) How varied are the ages of parents?**

Use STANDARD DEVIATION and RANGE — both measure how spread out or varied the data is.

- Range gives the spread between youngest and oldest parent.
- Standard Deviation gives a more precise measure of variability.

## Chapter 8: Database Concepts

*This chapter introduces database management systems (DBMS), the relational data model, and keys used to maintain data integrity.*

### Q1. Give the terms for each of the following:

**Answer:**

- a) Collection of logically related records—~~D~~ATABASE
- b) DBMS file containing description of data stored—~~D~~ATA DICTIONARY / DATABASE CATALOG / SCHEMA
- c) Attribute that can uniquely identify tuples in a relation—~~P~~RIMARY KEY (or CANDIDATE KEY)
- d) Special value stored when actual data value is unknown—~~N~~ULL
- e) Attribute that can uniquely identify tuples but is NOT defined as primary key—~~A~~LTERNATE KEY
- f) Software used to create, manipulate and maintain a relational database—~~R~~DBMS (Relational Database Management System)

### Q2. Why are foreign keys allowed to have NULL values? Explain with an example.

**Answer:**

Foreign keys reference the primary key of another table, establishing a relationship. NULL values are allowed in foreign keys because:

A NULL value in a foreign key means 'no relationship exists' for that record. It does not violate referential integrity because it simply means the record is not currently associated with any record in the referenced table.

**Example:**

Table: STUDENT(StudentID, Name, GuardianID)

Table: GUARDIAN(GuardianID, GuardianName, Phone)

GuardianID in STUDENT is a foreign key. If a student is an orphan or guardian details are not yet collected, GuardianID can be NULL — the student still exists but has no associated guardian record.

### Q3. Differentiate between:

**Answer:**

#### a) Database State vs Database Schema:

| Aspect      | Database State                                           | Database Schema                        |
|-------------|----------------------------------------------------------|----------------------------------------|
| Definition  | The actual data stored at a particular moment            | The structure/design of the database   |
| Changes     | Changes frequently (as data is inserted/updated/deleted) | Rarely changes                         |
| Also called | Database instance / snapshot                             | Intension or database description      |
| Example     | Current student records in the table                     | Table columns, data types, constraints |

#### b) Primary Key vs Foreign Key:

| Aspect | Primary Key | Foreign Key |
|--------|-------------|-------------|
|--------|-------------|-------------|

|                  |                                                 |                                               |
|------------------|-------------------------------------------------|-----------------------------------------------|
| Purpose          | Uniquely identifies each tuple in its own table | Links one table to the primary key of another |
| NULL allowed     | NO – must have a value                          | YES – can be NULL                             |
| Uniqueness       | Must be unique within the table                 | Can have duplicate values                     |
| Number per table | Only one primary key                            | Can have multiple foreign keys                |

### c) Degree vs Cardinality:

| Aspect     | Degree                                       | Cardinality                                |
|------------|----------------------------------------------|--------------------------------------------|
| Definition | Number of ATTRIBUTES (columns) in a relation | Number of TUPLES (rows) in a relation      |
| Represents | Width of the table                           | Height of the table                        |
| Example    | STUDENT(ID, Name, Age, Class) → Degree = 4   | Table has 500 students → Cardinality = 500 |

### Q4. How does DBMS avoid redundancy compared to file system?

#### Answer:

File systems store data independently in separate files, often duplicating the same information across multiple files. For example, a student's name might be stored in the attendance file, result file, and library file separately.

DBMS avoids this through:

- Normalization: Dividing data into related tables where each piece of data is stored only ONCE.
- Foreign Keys: Tables reference each other using keys instead of duplicating data.
- Central Repository: All data is managed from one system, preventing duplicate updates.

### Q5. What limitations of file system can be overcome by relational DBMS?

#### Answer:

1. Data Redundancy: Same data stored multiple times—DBMS eliminates through normalization.
2. Data Inconsistency: When redundant data is updated in one file but not others—DBMS ensures consistency.
3. Difficult Data Access: Custom programs needed for each query—DBMS provides SQL for easy querying.
4. No Concurrency Control: Multiple users accessing same file cause conflicts—DBMS handles concurrent access.
5. No Security: File system has limited access control—DBMS provides user-level permissions.
6. Difficult Backup/Recovery: Manual backups needed—DBMS has built-in backup and recovery features.

### Q6. Sports Preferences table — answer questions about relational properties.

#### Answer:

#### a) Can Roll No. 24 have NULL preference?

YES. The Preference column can have NULL because NULL means 'data unknown or not applicable'. Since Roll\_no 24 exists but has no sports preference, NULL is appropriate.

#### b) Roll No. 17 has two preferences — which property is violated?

This violates the ENTITY INTEGRITY rule — each student should have ONE unique tuple. Having two rows with Roll\_no 17 means Roll\_no cannot serve as a primary key.

Solution: Define Roll\_no as PRIMARY KEY. This will reject duplicate entries for the same roll number.

**c) Can Kabaddi (not chosen by any student) be in the relation?**

If NULL is allowed in Roll\_no, then a tuple (NULL, Kabaddi) could theoretically exist. However, if Roll\_no is the primary key, NULL is not allowed, so this tuple CANNOT be added.

**Q7. Are the states of both Sports Preferences relations equivalent?**

**Answer:**

YES, the states of both relations are equivalent. Here is why:

- Both tables contain the SAME data: same Roll\_no and Sports pairs.
- Table 1 is sorted by Roll\_no. Table 2 is sorted by Sports name.
- In relational database theory, the ORDER of tuples (rows) does NOT matter — a relation is a SET of tuples.
- Similarly, the ORDER of attributes (columns) also does NOT matter.
- Both tables have the same 5 tuples with identical data — they are mathematically equivalent relations.

**Q8. Design a canteen database — answer the design questions.**

**Answer:**

**a) Relation to store item name and price:**

```
ITEM(ItemCode INT PRIMARY KEY, ItemName VARCHAR(30) NOT NULL, Price FLOAT NOT NULL)
```

Each item stored only ONCE. PRIMARY KEY on ItemCode ensures uniqueness. NOT NULL on ItemName and Price.

**b) Relation for billing (quantity of item purchased):**

```
BILL(BillNo INT PRIMARY KEY, ItemCode INT, Quantity INT, BillDate DATE,
      FOREIGN KEY (ItemCode) REFERENCES ITEM(ItemCode))
```

BillNo is PRIMARY KEY (each bill is unique — addresses restriction i).

ItemCode as FOREIGN KEY ensures bill can only be for items in ITEM table (restriction ii).

**c) Where should 'calories' attribute be stored?**

'Calories' should be stored in the ITEM relation, since calories is a property of the item itself, not of a purchase. Add: ITEM(ItemCode, ItemName, Price, Calories)

**Q9. EMP-DEPENDENT database — answer questions about keys.**

**Answer:**

**Given:**

```
EMPLOYEE(AadharNumber, Name, Address, Department, EmployeeID)
DEPENDENT(EmployeeID, DependentName, Relationship)
```

**a) Candidate keys of EMPLOYEE:**

AadharNumber — unique for every individual in India.

EmployeeID — unique within the organization.

Both can be candidate keys. One will be chosen as Primary Key (likely EmployeeID).

**b) Tables and key required to retrieve dependents of a specific employee:**

Tables: EMPLOYEE and DEPENDENT

Key: EmployeeID — it is the primary key in EMPLOYEE and a foreign key in DEPENDENT, linking both tables.

**c) Degree of EMPLOYEE and DEPENDENT:**

EMPLOYEE has 5 attributes—Degree = 5

DEPENDENT has 3 attributes—Degree = 3

**Q10. SCHOOL\_UNIFORM database — can these tuples be inserted?****Answer:****a) Can these tuples be inserted into UNIFORM (UCode, UName, UColor)?**

i) (7, Handkerchief, NULL)—YES. UColor has no NOT NULL constraint, so NULL is allowed. UCode=7 is a new, valid primary key.

ii) (4, Ribbon, Red)—NO. UCode=4 already exists (Tie). Primary key must be unique. This would violate primary key constraint.

iii) (8, NULL, White)—NO. UName has NOT NULL constraint, so NULL is not allowed for UName.

**b) Can these tuples be inserted into COST (UCode, Size, Price)?**

i) (7, S, 0)—NO. The constraint states Price > 0. A price of 0 violates this constraint.

ii) (9, XL, 100)—NO. UCode=9 does not exist in the UNIFORM table. Foreign key constraint is violated — cannot insert a COST record for an item that doesn't exist in UNIFORM.

**Q11. Movie/Audi database — identify primary and foreign keys.****Answer:****Given:**

Movie(Movie\_ID, MovieName, ReleaseDate)

Audi(AudiNo, Movie\_ID, Seats, ScreenType, TicketPrice)

**Primary Keys:**

Movie—Movie\_ID (uniquely identifies each movie)

Audi—AudiNo (uniquely identifies each auditorium)

**Foreign Key:**

Movie\_ID in Audi references Movie\_ID in Movie. This links each auditorium to the movie being screened in it.

*One movie can appear in MULTIPLE rows in Audi (one per auditorium showing it), demonstrating a one-to-many relationship.*

## Chapter 9: Structured Query Language (SQL)

SQL is the standard language for managing and querying relational databases. This chapter covers DDL, DML, DQL commands, functions, GROUP BY, and JOIN operations.

### Q1. Answer the following questions:

#### Answer:

#### a) Define RDBMS. Name any two RDBMS software.

RDBMS (Relational Database Management System) is software that stores data in the form of related tables (relations). It supports creation, manipulation, and retrieval of data using SQL.

Examples: MySQL, Oracle, Microsoft SQL Server, PostgreSQL

#### b) Purpose of ORDER BY and GROUP BY:

ORDER BY: Sorts the result of a SELECT query in ascending (ASC) or descending (DESC) order based on specified columns.

GROUP BY: Groups rows that have the same value in specified columns into summary rows (used with aggregate functions like COUNT, SUM, AVG).

#### c) Differences between Single Row Functions and Aggregate Functions:

| Aspect             | Single Row Functions                | Aggregate Functions                     |
|--------------------|-------------------------------------|-----------------------------------------|
| Operation          | Operate on each row individually    | Operate on a group/set of rows          |
| Output             | Returns one result per input row    | Returns ONE result for the entire group |
| Examples           | UPPER(), LOWER(), LENGTH(), ROUND() | COUNT(), SUM(), AVG(), MAX(), MIN()     |
| Used with GROUP BY | Not necessary                       | Often used with GROUP BY                |

#### d) Cartesian Product:

Also called CROSS JOIN. It combines every row of one table with every row of another table. If Table A has m rows and Table B has n rows, the result has  $m \times n$  rows.

#### e) Differences:

##### ALTER vs UPDATE:

| ALTER                                 | UPDATE                                  |
|---------------------------------------|-----------------------------------------|
| DDL command – changes table STRUCTURE | DML command – changes table DATA/values |
| Adds/modifies/drops columns           | Changes values in rows                  |
| Example: ALTER TABLE t ADD col INT;   | Example: UPDATE t SET col=5 WHERE id=1; |

##### DELETE vs DROP:

| DELETE                                 | DROP                                         |
|----------------------------------------|----------------------------------------------|
| DML – removes specific ROWS from table | DDL – removes the entire TABLE from database |
| Table structure remains intact         | Table structure is also deleted              |

Can use WHERE clause

Cannot be reversed without backup

**f) Functions for operations:**

- i) Day of week from date (Independence Day) ~~DAYNAME('1947-08-15')~~
- ii) Specific characters from a position in string ~~SUBSTR()~~ or MID()
- iii) Name of month from birthdate ~~MONTHNAME(date)~~
- iv) Name in capital letters ~~UPPER(name)~~

**Q2. Write output of the following SQL statements:****Answer:**

```

SELECT POW(2,3);           -- Output: 8   (2 raised to power 3)
SELECT ROUND(342.9234,-1); -- Output: 340 (rounds to nearest 10)
SELECT LENGTH('Informatics Practices'); -- Output: 22
SELECT YEAR('1979/11/26'), MONTH('1979/11/26'), DAY('1979/11/26'),
MONTHNAME('1979/11/26');
-- Output: 1979 | 11 | 26 | November
SELECT LEFT('INDIA',3), RIGHT('Computer Science',4), MID('Informatics',3,4),
SUBSTR('Practices',3);
-- LEFT: IND | RIGHT: ence | MID: form | SUBSTR: actices

```

**Q3. SQL queries on MOVIE table:****Answer:**

```

-- a) Display all information
SELECT * FROM Movie;

-- b) MovieID, MovieName and Total_Earning
SELECT MovieID, MovieName, (ProductionCost + BusinessCost) AS Total_Earning
FROM Movie;

-- c) Different categories
SELECT DISTINCT Category FROM Movie;

-- d) Net profit (BusinessCost - ProductionCost)
SELECT MovieID, MovieName, (BusinessCost - ProductionCost) AS Net_Profit FROM
Movie;

-- e) ProductionCost between 10000 and 100000
SELECT MovieID, MovieName, ProductionCost AS Cost
FROM Movie WHERE ProductionCost > 10000 AND ProductionCost < 100000;

-- f) Comedy or Action movies
SELECT * FROM Movie WHERE Category IN ('Comedy', 'Action');

-- g) Movies not yet released (ReleaseDate is NULL)
SELECT * FROM Movie WHERE ReleaseDate IS NULL;

```

**Q4. Create Sports database, TEAM and MATCH\_DETAILS tables:****Answer:**

```

-- a) Create database
CREATE DATABASE Sports;
USE Sports;

-- b) Create TEAM table

```

```

CREATE TABLE TEAM (
    TeamID INT CHECK (TeamID BETWEEN 1 AND 9),
    TeamName VARCHAR(20) NOT NULL
);

-- c) Add primary key constraint
ALTER TABLE TEAM ADD CONSTRAINT pk_team PRIMARY KEY (TeamID);

-- d) Show structure
DESC TEAM;

-- e) Insert rows
INSERT INTO TEAM VALUES (1, 'Team Titan');
INSERT INTO TEAM VALUES (2, 'Team Rockers');
INSERT INTO TEAM VALUES (3, 'Team Magnet');
INSERT INTO TEAM VALUES (4, 'Team Hurricane');

-- f) Show contents
SELECT * FROM TEAM;

-- g) Create MATCH_DETAILS table
CREATE TABLE MATCH_DETAILS (
    MatchID INT PRIMARY KEY,
    MatchDate DATE,
    FirstTeam INT,
    SecondTeam INT,
    FirstTeam_Score INT,
    SecondTeam_Score INT,
    Winner INT,
    FOREIGN KEY (FirstTeam) REFERENCES TEAM(TeamID),
    FOREIGN KEY (SecondTeam) REFERENCES TEAM(TeamID),
    FOREIGN KEY (Winner) REFERENCES TEAM(TeamID)
);

```

### Q5. Queries on MATCH\_DETAILS:

#### Answer:

```

-- a) Matches where BOTH teams scored > 70
SELECT MatchID FROM MATCH_DETAILS
WHERE FirstTeam_Score > 70 AND SecondTeam_Score > 70;

-- b) FirstTeam < 70 but SecondTeam > 70
SELECT MatchID FROM MATCH_DETAILS
WHERE FirstTeam_Score < 70 AND SecondTeam_Score > 70;

-- c) Matches played by Team 1 AND won by Team 1
SELECT MatchID, MatchDate FROM MATCH_DETAILS
WHERE (FirstTeam=1 OR SecondTeam=1) AND Winner=1;

-- d) Matches played by Team 2 but NOT won by it
SELECT MatchID FROM MATCH_DETAILS
WHERE (FirstTeam=2 OR SecondTeam=2) AND (Winner != 2 OR Winner IS NULL);

-- e) Rename table and columns
ALTER TABLE TEAM RENAME TO T_DATA;

```

```
ALTER TABLE T_DATA CHANGE TeamID T_ID INT;
ALTER TABLE T_DATA CHANGE TeamName T_NAME VARCHAR(20);
```

## Q6. Wonderful Garments SCHOOLUNIFORM database — rectify anomalies:

### Answer:

```
-- a) Add handkerchief to UNIFORM table first
INSERT INTO UNIFORM (UCode, UName, UColor) VALUES (7, 'Handkerchief', 'Red');
-- Then add to COST table
INSERT INTO COST (UCode, Size, Price) VALUES (7, 'M', 100);

-- b) Add FOREIGN KEY constraint to prevent orphan COST records
ALTER TABLE COST
ADD CONSTRAINT fk_ucode FOREIGN KEY (UCode) REFERENCES UNIFORM(UCode);
-- Now the problematic insert (7, 'M', 100) without first inserting into
UNIFORM will be rejected.

-- c) Add NOT NULL constraint on UName
ALTER TABLE UNIFORM MODIFY UName VARCHAR(30) NOT NULL;
-- This ensures UName must have a valid value before a UCode can be assigned.
```

## Q7 (Product table). SQL queries for Product table:

### Answer:

*Assumed schema: Product(PCode, PName, Manufacturer, UPrice)*

```
-- a) Create Product table
CREATE TABLE Product (
    PCode    INT PRIMARY KEY,
    PName    VARCHAR(30) NOT NULL,
    Manufacturer VARCHAR(30),
    UPrice    FLOAT CHECK (UPrice > 0)
);

-- b) Primary key is PCode

-- c) List PCode, PName, UPrice descending by PName, then ascending by price
SELECT PCode, PName, UPrice FROM Product
ORDER BY PName DESC, UPrice ASC;

-- d) Add Discount column
ALTER TABLE Product ADD Discount FLOAT DEFAULT 0;

-- e) Set discount: 10% if UPrice > 100, else 0
UPDATE Product SET Discount = CASE
    WHEN UPrice > 100 THEN UPrice * 0.10
    ELSE 0
END;

-- f) Increase price by 12% for Dove products
UPDATE Product SET UPrice = UPrice * 1.12 WHERE Manufacturer = 'Dove';

-- g) Total products per manufacturer
SELECT Manufacturer, COUNT(*) AS TotalProducts
FROM Product GROUP BY Manufacturer;
```

```
-- h) AVG UPrice grouped by PName
SELECT PName, AVG(UPrice) FROM Product GROUP BY PName;

-- i) Distinct manufacturers
SELECT DISTINCT Manufacturer FROM Product;

-- j) Count of distinct PNames
SELECT COUNT(DISTINCT PName) FROM Product;

-- k) MAX and MIN price per PName
SELECT PName, MAX(UPrice), MIN(UPrice) FROM Product GROUP BY PName;
```

#### **Q8. CARSHOWROOM database — add Discount column, set values, display costliest car:**

##### **Answer:**

*Assumed INVENTORY schema: INVENTORY(CName, Model, MFYear, Price, Colour)*

```
-- a) Add Discount column
ALTER TABLE INVENTORY ADD Discount FLOAT DEFAULT 0;

-- b) Set discount based on model
UPDATE INVENTORY SET Discount = CASE
    WHEN Model = 'LXI' THEN 0
    WHEN Model = 'VXI' THEN Price * 0.10
    ELSE Price * 0.12
END;

-- c) Display name of costliest car
SELECT CName FROM INVENTORY WHERE Price = (SELECT MAX(Price) FROM INVENTORY);
```

## Chapter 10: Computer Networks

*This chapter covers the fundamentals of computer networking — types of networks, networking devices, topologies, IP/MAC addressing, and the Internet.*

### Q1. Expand the following:

#### Answer:

- a) ARPANET – Advanced Research Projects Agency NETwork
- b) MAC – Media Access Control
- c) ISP – Internet Service Provider
- d) URI – Uniform Resource Identifier

### Q2. What do you understand by the term network?

#### Answer:

A network is a collection of two or more computers (or devices) connected together to share resources, exchange data, and communicate. Networks allow users to share files, printers, internet connections, and software applications.

### Q3. Mention any two main advantages of using a network of computing devices.

#### Answer:

1. Resource Sharing: Hardware resources like printers, scanners, and storage can be shared among multiple users, reducing costs.
2. Communication: Users can communicate via email, messaging, and video calls. Data can be transferred quickly between computers.

### Q4. Differentiate between LAN and WAN.

#### Answer:

| Feature   | LAN (Local Area Network)            | WAN (Wide Area Network)                   |
|-----------|-------------------------------------|-------------------------------------------|
| Coverage  | Small area (room, building, campus) | Large area (cities, countries, worldwide) |
| Speed     | High speed (100 Mbps to 10 Gbps)    | Lower speed (due to large distance)       |
| Cost      | Relatively inexpensive              | Expensive to set up and maintain          |
| Ownership | Usually owned by one organization   | Often owned by telecom/service providers  |
| Example   | Office network, school lab network  | The Internet, banking network             |

### Q5. Names of commonly used networking devices:

#### Answer:

Router, Switch, Hub, Repeater, Bridge, Gateway, Modem, NIC (Network Interface Card), Access Point

### Q6. Two universities in different States want to transfer information — which type of network?

#### Answer:

They need a WAN (Wide Area Network) or can use the Internet. Since they are in different states (large geographic distance), a LAN won't cover the distance. WAN/Internet is used for such inter-state or inter-city communication.

### Q7. Define topology. What are popular network topologies?

#### Answer:

Network topology refers to the physical or logical arrangement of nodes (computers/devices) and connections (cables/links) in a network.

Popular topologies:

- Bus Topology: All devices connected to a single cable.
- Star Topology: All devices connected to a central hub/switch.
- Ring Topology: Devices connected in a circular loop.
- Mesh Topology: Every device connected to every other device.
- Tree Topology: Hierarchical arrangement combining bus and star.

### Q8. How is tree topology different from bus topology?

#### Answer:

| Feature        | Bus Topology                              | Tree Topology                                     |
|----------------|-------------------------------------------|---------------------------------------------------|
| Structure      | All nodes connected to ONE central cable  | Hierarchical – nodes connected like tree branches |
| Branching      | No branching                              | Multiple branches/levels                          |
| Failure impact | Entire network fails if main cable breaks | Only the affected branch fails                    |
| Scalability    | Difficult to extend                       | Easy to extend by adding branches                 |

### Q9. Identify topology type:

#### Answer:

- Each node is connected with the help of a single cable—**BUS Topology**
- Each node is connected with central switching through independent cables—**STAR Topology**

### Q10. What is a modem? Why is it used?

#### Answer:

Modem stands for **MO**dulator-**DE**Modulator. It is a networking device that converts digital signals (from a computer) to analog signals (for telephone/cable lines) and vice versa.

It is used to connect a home or office network to the Internet through ISPs via telephone lines, cable, or fiber. Without a modem, a computer cannot communicate over analog communication channels.

### Q11. Explain networking devices:

#### Answer:

#### a) Switch:

A switch connects multiple devices in a LAN. Unlike a hub, it sends data **ONLY** to the specific device it is addressed to (using MAC addresses), making it more efficient.

#### b) Repeater:

A repeater amplifies or regenerates a weakening signal over long distances, allowing data to travel further without degrading.

#### c) Router:

A router connects different networks (e.g., your home network to the Internet). It determines the best path for data packets to reach their destination using IP addresses.

**d) Gateway:**

A gateway acts as an entry/exit point between two different types of networks (e.g., LAN to WAN). It can translate between different protocols.

**e) NIC (Network Interface Card):**

NIC is a hardware component (circuit board) installed in a computer that enables it to connect to a network. It has a unique MAC address.

**Q12. Draw a network layout of star and bus topology connecting five computers.****Answer:**

[Diagram prompt: Star Topology — Draw a central switch/hub in the middle with 5 computers (C1-C5) connected to it like spokes of a wheel. Bus Topology — Draw a single horizontal cable (backbone) with all 5 computers connected vertically to it at different points, with terminators at both ends]

**Q13. What is the significance of MAC address?****Answer:**

MAC (Media Access Control) address is a unique hardware identifier assigned to every Network Interface Card (NIC) by its manufacturer. It is a 48-bit (6-byte) address usually written in hexadecimal format (e.g., 00:1A:2B:3C:4D:5E).

Significance:

- Used for data delivery within a local network (LAN layer)
- Permanently embedded in hardware — cannot be changed (though it can be spoofed)
- Switches use MAC addresses to forward frames to the correct device

**Q14. How is IP address different from MAC address?****Answer:**

| Feature     | IP Address                         | MAC Address                          |
|-------------|------------------------------------|--------------------------------------|
| Full form   | Internet Protocol address          | Media Access Control address         |
| Type        | Logical (software-assigned)        | Physical (hardware-based)            |
| Changes     | Can change (dynamic IP)            | Permanently assigned (usually fixed) |
| Format      | 32-bit (IPv4) e.g., 192.168.1.1    | 48-bit hex e.g., 00:1A:2B:3C:4D:5E   |
| Used for    | Routing across networks (Internet) | Addressing within LAN                |
| Assigned by | Network admin/ISP/DHCP server      | Manufacturer of NIC                  |

**Q15. What is DNS? What is a DNS server?****Answer:**

DNS (Domain Name System) is a system that translates human-readable domain names (like www.google.com) into machine-readable IP addresses (like 142.250.80.46). This allows users to type easy-to-remember names instead of numbers.

DNS Server: A DNS server is a computer that stores a database of domain names and their corresponding IP addresses. When you type a URL in a browser, the DNS server looks it up and returns the correct IP address so your browser can connect.

**Q16. Help Sahil understand World Wide Web vs Internet:**

**Answer:****Internet:**

The Internet is the global PHYSICAL infrastructure — a massive network of interconnected computers, servers, routers, and cables spanning the world. It supports various services.

Example: The Internet includes email, FTP, online gaming, VoIP calls, and the Web.

**World Wide Web (WWW):**

The Web is one SERVICE that runs ON TOP of the Internet. It is a system of interlinked hypertext documents and media accessible via web browsers using HTTP/HTTPS.

Example: When you open Chrome and visit a website like Wikipedia — that is the Web.

**Analogy: The Internet is like the road system; the Web is like one type of vehicle (cars) that uses those roads. Email, FTP, etc. are other vehicles.**

# Chapter 11: Data Communication

*Data communication involves the exchange of data between devices through a transmission medium. This chapter covers components, bandwidth, transmission media, protocols, and mobile technologies.*

## Q1. What is data communication? What are the main components?

### Answer:

Data communication refers to the electronic exchange of data between two or more devices through a transmission medium (wired or wireless). The data can be in the form of text, numbers, images, audio, or video.

### Five main components:

1. Sender: The device that sends the data (computer, phone, etc.)
2. Receiver: The device that receives the data
3. Message: The data/information being communicated (text, image, video, etc.)
4. Transmission Medium: The physical path through which data travels (cable, air, optical fiber)
5. Protocol: The set of rules that govern how data is sent and received

*[Diagram: Refer to Figure 11.2 in textbook showing Sender → Transmission Media → Receiver with protocol governing the exchange]*

## Q2. Which communication medium is used for satellite communication?

### Answer:

Satellite communication uses MICROWAVE (wireless) as the communication medium, specifically through communication satellites orbiting the Earth. Microwaves travel in straight lines (line-of-sight), so satellites are used to relay signals over long distances where direct line-of-sight is not possible.

## Q4. What are three categories of wired media? Explain them.

### Answer:

#### 1. Twisted Pair Cable:

Two insulated copper wires twisted together to reduce electromagnetic interference. Used in telephone lines and Ethernet networks (LAN). Types: UTP (Unshielded) and STP (Shielded).

#### 2. Coaxial Cable:

A central conductor surrounded by insulation, a metallic shield, and an outer cover. Provides better bandwidth and interference resistance than twisted pair. Used in cable TV and broadband Internet.

#### 3. Optical Fiber Cable:

Transmits data as pulses of light through thin glass or plastic fibers. Extremely high bandwidth, immune to electromagnetic interference, and can carry signals over very long distances. Used for submarine cables, backbone networks.

## Q5. Compare wired and wireless media.

### Answer:

| Feature      | Wired Media                          | Wireless Media                             |
|--------------|--------------------------------------|--------------------------------------------|
| Transmission | Uses physical cables                 | Uses electromagnetic waves                 |
| Speed        | Generally faster and more reliable   | Slower, affected by obstacles/interference |
| Security     | More secure (physical access needed) | Less secure (signals can be intercepted)   |

|              |                                       |                                        |
|--------------|---------------------------------------|----------------------------------------|
| Installation | Difficult, requires cables to be laid | Easy, flexible mobility                |
| Cost         | Higher initial cost (cabling)         | Lower initial cost                     |
| Examples     | Optical fiber, coaxial, twisted pair  | Wi-Fi, Bluetooth, satellite, microwave |

#### Q6. Which transmission media carries signals in the form of light?

##### Answer:

Optical Fiber Cable carries signals in the form of light pulses. Data is transmitted as pulses of light (laser or LED) through thin strands of glass or plastic fiber using the principle of total internal reflection.

#### Q7. List advantages and disadvantages of optical fiber cable.

##### Answer:

##### Advantages:

- Very high bandwidth — can carry enormous amounts of data
- Not affected by electromagnetic interference
- Low signal attenuation — data can travel very long distances without repeaters
- Secure — difficult to tap without detection
- Lightweight and thin

##### Disadvantages:

- Expensive to install and maintain
- Fragile — glass fibers can break easily
- Difficult to splice or repair
- Requires specialized equipment for installation

#### Q8. What is the range of frequency for radio waves?

##### Answer:

Radio waves have a frequency range of 3 kHz to 1 GHz (3,000 Hz to 1,000,000,000 Hz). They are used for AM/FM radio, TV broadcasting, mobile phones (cellular), and satellite communication.

#### Q9. 18 Gbps is equal to how many Bits per second?

##### Answer:

$$\begin{aligned}
 18 \text{ Gbps} &= 18 \times 10^9 \text{ bps} \\
 &= 18,000,000,000 \text{ bps} \\
 &= 18 \times 1,000,000,000 \text{ bits per second}
 \end{aligned}$$

**Answer: 18 Gbps = 18,000,000,000 bits per second (18 billion bps)**

#### Q10. HTTP stands for?

##### Answer:

HTTP stands for HyperText Transfer Protocol. It is the primary protocol used to transfer web pages and resources on the World Wide Web. The secure version is HTTPS (HTTP Secure), which uses encryption (SSL/TLS).

#### Q11. Write short notes on:

##### Answer:

a) HTTP:

HTTP (HyperText Transfer Protocol) is the foundation protocol of data communication on the World Wide Web. It defines how web browsers (clients) request web pages from web servers and how servers respond. Developed by Tim Berners-Lee in 1989 at CERN.

**b) Bandwidth:**

Bandwidth is the range of frequencies available for transmission through a channel. It is measured in Hertz (Hz). Higher bandwidth means more data can be transmitted per unit time. E.g., a channel with frequencies from 100 Hz to 1100 Hz has bandwidth of 1000 Hz.

**c) Bluetooth:**

Bluetooth is a short-range wireless communication technology operating at 2.4 GHz frequency, with a range of about 10 meters. It connects devices like phones, keyboards, headphones, and computers wirelessly without cables.

**d) DNS (see Chapter 10, Q15):**

DNS translates domain names to IP addresses, enabling humans to use easy-to-remember website names.

**e) Data Transfer Rate:**

Data Transfer Rate (DTR) is the amount of data transmitted from one device to another in a given time period. Measured in bits per second (bps), Kbps, Mbps, or Gbps. Higher DTR means faster transmission.

**Q12. What is protocol in data communication? Explain with an example.**

**Answer:**

A protocol is a set of standardized rules and procedures that govern how data is transmitted, received, and acknowledged between communicating devices. Without protocols, devices from different manufacturers could not communicate.

Example — TCP/IP: When you open a website, your browser follows HTTP to request the page; TCP breaks the data into packets; IP addresses and routes each packet to the correct destination; at the receiver, TCP reassembles the packets in order.

**Q13. A composite signal contains frequencies between 500 MHz and 1 GHz. What is the bandwidth?**

**Answer:**

$$\begin{aligned}
 \text{Bandwidth} &= \text{Maximum frequency} - \text{Minimum frequency} \\
 &= 1 \text{ GHz} - 500 \text{ MHz} \\
 &= 1000 \text{ MHz} - 500 \text{ MHz} \\
 &= 500 \text{ MHz}
 \end{aligned}$$

**Bandwidth = 500 MHz**

## Chapter 12: Security Aspects

*This chapter covers network security threats, malware, antivirus, firewalls, encryption (HTTP vs HTTPS), cookies, hacking, and network intrusion techniques.*

### Q1. Why is a computer considered safe if not connected to a network or Internet?

#### Answer:

A standalone (isolated) computer faces NO external network threats because:

- Viruses and worms cannot spread to it through the network
- Hackers cannot remotely access it (no network connection)
- Ransomware cannot be delivered through phishing emails
- Spyware cannot transmit stolen data back to attackers

However, it is still vulnerable to physical threats (USB drives, stolen hardware) and internal threats. The 'air gap' (being disconnected) is used by governments and defence organisations for highly sensitive computers.

### Q2. What is a computer virus? Name some popular ones.

#### Answer:

A computer virus is malicious software code that attaches itself to legitimate programs and replicates when the infected program is run. It can corrupt files, slow down the system, display unwanted messages, or steal data. The term was coined by Fred Cohen in 1985.

#### Famous computer viruses:

- ILOVEYOU (2000) — spread via email, infected millions of computers worldwide
- Conficker (2008) — infected millions of Windows machines
- Melissa (1999) — mass-mailing virus
- CryptoLocker (2013) — ransomware virus encrypting user files
- WannaCry (2017) — ransomware that affected hospitals and corporations globally

### Q3. How is a computer worm different from a virus?

#### Answer:

| Feature     | Computer Virus                            | Computer Worm                                     |
|-------------|-------------------------------------------|---------------------------------------------------|
| Replication | Needs a HOST program to attach to         | Self-replicates independently – no host needed    |
| Spreading   | Spreads when infected file/program is run | Spreads automatically through network connections |
| Goal        | Usually corrupts files or data            | Usually consumes bandwidth and system resources   |
| Examples    | ILOVEYOU, Melissa                         | Morris Worm, Conficker                            |

### Q4. How is Ransomware used to extract money from users?

#### Answer:

Ransomware is a type of malware that:

1. Secretly installs itself on the victim's computer (via email attachment, malicious website, etc.)
2. Encrypts all important files (documents, photos, databases) making them inaccessible

3. Displays a ransom note demanding payment (usually in cryptocurrency like Bitcoin) to receive the decryption key
4. Threatens to permanently delete files or publish sensitive data if payment is not made within a deadline

Example: WannaCry (2017) demanded Bitcoin payments and crippled the UK's National Health Service.

#### Q5. How did a Trojan get its name?

##### Answer:

A Trojan Horse (in computing) gets its name from the famous ancient Greek story of the Trojan War. In the story, Greek soldiers hid inside a giant wooden horse gifted to Troy. The Trojans unsuspectingly brought the 'gift' inside their city walls, and at night the soldiers emerged to attack from within.

Similarly, a computer Trojan appears as a LEGITIMATE or useful program (a game, utility, or free software), but secretly contains malicious code. When the user installs it, the malware runs hidden in the background — just like soldiers hidden in the horse.

#### Q6. How does adware generate revenue for its creator?

##### Answer:

Adware (advertising-supported software) generates revenue through:

- Displaying advertisements in the user's browser or application — creators are paid per ad view or click (Pay-Per-Click model)
- Collecting browsing data and selling it to advertisers for targeted advertising
- Redirecting users to specific websites, increasing traffic and generating affiliate commissions
- Installing browser toolbars that display sponsored search results

#### Q7. Briefly explain two threats from a keylogger installed on a computer.

##### Answer:

##### Threat 1 — Password Theft:

A keylogger records every keystroke, including passwords typed for banking, email, and social media accounts. The attacker receives these logs and can use the stolen credentials to gain unauthorized access.

##### Threat 2 — Financial Fraud:

Keyloggers can capture credit card numbers, CVV codes, and internet banking details when typed during online transactions. This enables the attacker to commit financial fraud and unauthorized transactions.

#### Q8. How is Virtual Keyboard safer than On Screen Keyboard?

##### Answer:

##### On Screen Keyboard (OSK):

The OSK is part of the operating system and generates the same keyboard events (signals) as a physical keyboard. A keylogger can still intercept these signals.

##### Virtual Keyboard:

A virtual keyboard (provided by banking websites, security software) uses a different method — it often randomizes the layout of keys each time, uses mouse clicks (not keystrokes), and may use screen capture protection. This makes it much harder for keyloggers (which record key events) to capture what was 'typed'.

#### Q9. List different modes of malware distribution.

##### Answer:

1. Email Attachments: Malicious files sent as email attachments (Word docs with macros, PDFs)

2. Malicious Websites: Visiting compromised websites that automatically download malware (drive-by downloads)
3. USB/Removable Drives: Infected USB drives auto-run malware when plugged in
4. Software Downloads: Fake or cracked software containing hidden malware
5. Social Engineering/Phishing: Tricking users to click malicious links
6. Network Propagation: Worms spreading through network vulnerabilities

#### Q10. List common signs of malware infection.

##### Answer:

- Computer is unusually slow or frequently crashes (BSOD)
- Unexpected pop-up advertisements
- Browser homepage changes without user's action
- Unusual network activity (data being sent/received)
- Programs opening/closing on their own
- Files being deleted or encrypted
- Hard drive light continuously active
- Antivirus disabled or unavailable

#### Q11. List some preventive measures against malware infection.

##### Answer:

- Install and regularly update antivirus/anti-malware software
- Keep operating system and applications patched/updated
- Do not open suspicious email attachments or click unknown links
- Download software only from official/trusted sources
- Use a firewall to monitor incoming/outgoing traffic
- Regularly back up important data
- Be cautious with USB drives from unknown sources

#### Q12. Write short note on methods of malware identification used by antivirus software.

##### Answer:

##### 1. Signature-Based Detection:

Compares files against a database of known malware 'signatures' (patterns). Effective for known malware but fails for new/modified variants.

##### 2. Heuristic Analysis:

Analyses code behaviour and characteristics to identify malware-like patterns even without an exact signature match. Can detect new/unknown malware.

##### 3. Behavioral Monitoring:

Monitors running programs for suspicious activities (unexpected file encryption, registry changes). Identifies malware by what it DOES rather than what it looks like.

#### Q13. What are risks associated with HTTP? How does HTTPS resolve them?

##### Answer:

##### Risks with HTTP:

- Data sent in plain text — anyone who intercepts the traffic can read it (passwords, credit card numbers)
- Man-in-the-Middle attacks: attackers can intercept and modify data between browser and server
- No authentication — users cannot verify if they are connected to the real website

**How HTTPS resolves these:**

HTTPS adds SSL/TLS encryption layer which: (1) encrypts all data in transit so interceptors see only gibberish, (2) authenticates the server using digital certificates verifying identity, (3) ensures data integrity so modifications are detected.

**Q14. List one advantage and disadvantage of Cookies.****Answer:****Advantage:**

Cookies store session information, user preferences, and login status — making websites personalized and convenient (e.g., 'Remember me' on login, shopping cart contents).

**Disadvantage:**

Third-party cookies track users across multiple websites, raising serious privacy concerns. This tracking data can be sold to advertisers without user's explicit consent.

**Q15. Write short note on White, Black, and Grey Hat Hackers.****Answer:****White Hat Hackers (Ethical Hackers):**

Security professionals who hack systems WITH PERMISSION from the owner to find and fix vulnerabilities. They work legally and help organizations improve security.

**Black Hat Hackers (Crackers):**

Malicious hackers who breach systems WITHOUT PERMISSION for personal gain, espionage, or disruption. Their activities are illegal.

**Grey Hat Hackers:**

Operate between white and black hat. They may hack without permission but inform the owner of vulnerabilities (sometimes demanding payment). Their actions are ethically ambiguous.

**Q16. Differentiate between DoS and DDoS attack.****Answer:**

| Feature   | DoS (Denial of Service)                         | DDoS (Distributed DoS)                              |
|-----------|-------------------------------------------------|-----------------------------------------------------|
| Source    | Single computer/attacker                        | Multiple computers (hundreds/thousands)             |
| Scale     | Smaller scale attack                            | Much larger, harder to stop                         |
| Blocking  | Can block the single IP                         | Cannot block all IPs (many sources)                 |
| Mechanism | Overwhelms server with requests from one source | Floods server from distributed zombie computers     |
| Example   | One computer sending millions of packets        | Botnet sending requests from infected PCs worldwide |

**Q17. How is Snooping different from Eavesdropping?****Answer:**

| Feature    | Snooping                               | Eavesdropping                          |
|------------|----------------------------------------|----------------------------------------|
| Definition | Secret capture AND analysis of network | Unauthorized REAL-TIME interception of |

|        |                                                        |                                                      |
|--------|--------------------------------------------------------|------------------------------------------------------|
|        | traffic by malicious users                             | communication between two parties                    |
| Nature | Can be passive recording of traffic for later analysis | Active real-time monitoring/listening                |
| Goal   | Collect data, passwords, or sensitive info over time   | Intercept private conversations or data in real time |

## Notes for Students

These solutions cover all exercises from Chapters 1 through 12 of the Class XII Computer Science NCERT textbook (2025-26 edition). Chapter 13 (Project Based Learning) is a practical project chapter and does not contain traditional exercise questions.

Tips for exam preparation:

- For programming questions: Focus on understanding the logic first, then code. Practice writing programs by hand.
- For database/SQL questions: Practise running queries on MySQL. Understanding is better than memorization.
- For theory questions (Networks, Security): Learn definitions precisely. Use bullet points in exam answers for clarity.
- For Stack/Queue: Practice trace-tables extensively — they are common exam questions.

**Good luck with your exams!**