Algorithms, Programming, and Logic

7. Algorithm Design and Problem-Solving:

- **Program Development Life Cycle:**
    - **Analysis:**
        - **Abstraction:** Focus on essential aspects, ignoring irrelevant details.
        - **Decomposition:** Break down the problem into smaller parts.
        - **Identification:** Determine the exact problem and requirements.
    - **Design:**
        - **Decomposition:** Further break down the problem into steps or modules.
        - **Structure Diagrams:** Visual representations of system architecture (e.g., data flow diagrams).
        - **Flowcharts:** Diagrams representing the flow of control through an algorithm.
        - **Pseudocode:** High-level description of an algorithm using simple language resembling code.
    - **Coding:**
        - **Write Program Code:** Translate the algorithm into a programming language.
        - **Iterative Testing:** Continuously test and refine the code to find and fix errors.
    - **Testing:**
        - **Test Data:** Use various data sets to ensure the program handles all scenarios (e.g., normal, boundary, extreme).
- **Standard Methods of Solution:**

Linear Search:
python
Copy code
```python
def linear_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1
```

○

Bubble Sort:
python
Copy code
```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
```

- ○
- **Validation and Verification:**
    - **Validation Checks:** Ensure data meets criteria before processing.
        - **Range Check:** Validates if a value is within a specified range (e.g., age between 0 and 120).
        - **Format Check:** Ensures data follows a specific format (e.g., email addresses).
    - **Verification Checks:** Confirm data entry accuracy.
        - **Visual Check:** Manually reviewing data.
        - **Double Entry Check:** Entering data twice and comparing results.
- **Test Data Types:**
    - **Normal:** Typical data values the program is expected to handle (e.g., valid user inputs).
    - **Abnormal:** Unexpected or erroneous values (e.g., negative age).
    - **Extreme:** Values at the boundary of acceptable limits (e.g., maximum integer value).
    - **Boundary:** Values at the edge of acceptable limits and just outside (e.g., maximum allowable age and one more).

8. Programming Concepts:

- **Basic Constructs:**

Variables and Constants:
python
Copy code
```
age = 25   # Variable
PI = 3.14  # Constant
```

- ○
- **Data Types:**
    - **Integer:** Whole numbers (`int`)
    - **Real:** Floating-point numbers (`float`)
    - **Char:** Single characters (`char`)
    - **String:** Text data (`string`)
    - **Boolean:** True/False values (`bool`)
- **Control Structures:**

        - **Sequence:** Direct execution of code statements.

Selection: Conditional execution of code (e.g., `if` statements).
python
Copy code
```
if age >= 18:
    print("Adult")
else:
    print("Minor")
```

        -

Iteration: Repeated execution of code (e.g., loops).
python
Copy code
```
for i in range(5):
    print(i)
```

        -
- **String Handling:**
    - **Operations:**
        - **Length:** `len(string)`
        - **Substring:** `string[start:end]`
        - **Upper/Lower Case:** `string.upper()`, `string.lower()`
- **Operators:**

Arithmetic Operators:
python
Copy code
```
result = 10 + 5  # Addition
result = 10 - 5  # Subtraction
result = 10 * 5  # Multiplication
result = 10 / 5  # Division
result = 10 % 3  # Modulus
```

    -
    - **Boolean Operators:**
        - **AND:** `True and False` results in `False`
        - **OR:** `True or False` results in `True`
        - **NOT:** `not True` results in `False`
- **Procedures and Functions:**

Procedures: Perform tasks without returning a value.
python
Copy code
```
def greet():
    print("Hello, World!")
```

-

Functions: Perform tasks and return a value.
python
Copy code
```
def add(a, b):
    return a + b
```

-

Parameters: Provide input to procedures/functions.
python
Copy code
```
def multiply(x, y):
    return x * y
```

-
- **Maintainable Programs:**
    - **Meaningful Identifiers:** Use descriptive names for variables, constants, and functions.
    - **Commenting:** Explain what different parts of the code do for clarity.

Arrays:

One-Dimensional (1D) Arrays:
python
Copy code
```
numbers = [1, 2, 3, 4, 5]
print(numbers[0])  # Access first element
```

-

Two-Dimensional (2D) Arrays:
python
Copy code
```
matrix = [[1, 2], [3, 4]]
print(matrix[0][1])  # Access element in first row, second column
```

-

File Handling:

- **Purpose:** Store and retrieve data from files.
- **Operations:**
    - **Open File:** `file = open("filename.txt", "r")`

Read/Write Data:
python
Copy code
```
with open("filename.txt", "w") as file:
    file.write("Hello, World!")
```

-
- **Close File:** `file.close()`

9. Databases:

- **Single-Table Database:**
    - **Fields:** Columns in a table (e.g., `Name`, `Age`).
    - **Records:** Rows in a table (e.g., data for each individual).
- **Primary Key:** Unique identifier for each record in a table.
- **SQL Basics:**

SELECT: Retrieve data.
sql
Copy code
```
SELECT * FROM table_name;
```

-
- **FROM:** Specify the table.

WHERE: Filter results.
sql
Copy code
```
SELECT * FROM table_name WHERE condition;
```

-

ORDER BY: Sort results.
sql
Copy code
```
SELECT * FROM table_name ORDER BY column_name;
```

-

10. Boolean Logic:

- **Logic Gates:**

AND Gate:
plaintext
Copy code
```
A B | Output
0 0 | 0
0 1 | 0
1 0 | 0
1 1 | 1
```

-

OR Gate:
plaintext
Copy code
```
A B | Output
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 1
```

-

NOT Gate:
plaintext
Copy code
```
A | Output
0 | 1
1 | 0
```

-
- **Creating and Understanding Logic Circuits:**
    - **Truth Tables:** Show possible input combinations and their outputs.