DS Programming Questions

Day	Questions	Understood	Implem
1 Intro to Programming	1. Write a Python program to calculate the factorial of a number.		
	2. Implement a Python program to find the sum of even numbers in a list.		
	3. Create a Python function to reverse a given string.		
	4. Write a program to check if a number is prime.		
	5. Develop a Python program to solve a basic math problem, e.g., finding the area of a rectangle.		
2 Variables, Data Types	1. Write a Python program to convert temperature from Fahrenheit to Celsius.		
	2. Implement a program to calculate the area of a circle.		
	3. Create a Python function to check if a given number is even or odd.		
	4. Write a program to find the square root of a number.		
	5. Develop a Python program to calculate the simple interest.		
3 Control Flow	1. Write a Python program to check if a year is a leap year.		
	2. Implement a program to find the maximum of three numbers.		
	3. Create a Python function to calculate the sum of natural numbers up to N.		
	4. Write a program to print the Fibonacci sequence up to N terms.		
	5. Develop a Python program to check if a number is a palindrome.		
4 Collections	1. Write a Python program to count the number of vowels in a string.		
	2. Implement a program to find the intersection of two lists.		
	3. Create a Python function to reverse a given list.		
	4. Write a program to count the frequency of elements in a list.		
	5. Develop a Python program to remove duplicates from a list.		
5 Functions	1. Implement a Python program to calculate the power of a number using a recursive function.		
	2. Write a program to find the LCM of two numbers using a function.		
	3. Create a Python function to generate a random password of a given length.		
	4. Write a program to find the GCD of two numbers using a function.		
	5. Develop a Python program to check if a string is a palindrome using a function.		
6 Functions	1. Implement a Python program to calculate the exponential of a number using a user-defined function.		
	2. Write a program to find the mean and median of a list of numbers using functions.		
	3. Create a Python function to check if a given string is a valid palindrome (ignoring spaces and punctuation).		
	4. Write a program to generate a Fibonacci sequence using a recursive function.		
	5. Develop a Python program to find the greatest common divisor (GCD) of two numbers using a recursive		
7 Scope of Variables and Recursion	1. Write a Python program to demonstrate the scope of variables in different functions.		
	2. Implement a program to calculate the sum of natural numbers using recursion.		
	3. Create a recursive function to find the factorial of a number.		
	4. Write a program to solve the Tower of Hanoi puzzle using recursion.		
	5. Develop a Python program to compute the nth Fibonacci number using recursion.		
8 Object-Oriented Programming (OOP) Basics	1. Implement a Python class representing a simple bank account with deposit and withdrawal methods.		
	2. Write a Python program to create an instance of a car object with attributes like make, model, and year.		
	3. Create a Python class for a basic calculator that can add, subtract, multiply, and divide.		
	4. Write a program to create a Python class representing a book with attributes like title and author.		
	5. Develop a Python program to demonstrate the concept of encapsulation using a class.		
9 OOP Concepts (Constructors, Class)	1. Implement a Python class representing a student with a constructor and methods for setting and getting student details.		
	2. Write a Python program to create a class diagram for a simple online shopping system.		
	3. Create a Python class for a basic shape with constructor and methods to calculate area and perimeter.		
	4. Develop a Python program that demonstrates the concept of inheritance with multiple classes.		
	5. Write a program to create a Python class representing a basic employee with attributes like name, id, and		
10 More OOP (Statics, Relationship, Inheritance, Abstract Classes)	1. Implement a Python class with a static variable to keep track of the number of instances created.		
	2. Write a Python program to create a class relationship diagram for a library system.		
	3. Create a Python class representing a geometric shape with methods for calculating area and perimeter.		
	4. Develop a Python program to demonstrate single and multiple inheritance with classes.		
	5. Write a program to create an abstract class representing a vehicle with abstract methods for moving and		
11 Exception Handling	1. Write a Python program that demonstrates the use of a try-except block to handle a division by zero error.		
	2. Implement a program that raises a custom exception when a user tries to withdraw more money from an ATM than the account balance.		
	3. Create a Python function that takes a list as input and handles an "Index out of Range" exception.		
	4. Develop a program that uses a try-except block to handle a FileNotFoundError when reading data from a file.		
	5. Write a Python program that demonstrates the use of multiple except blocks for different types of exceptions.		
12 Abstract Data Types (ADT)	1. Implement a Python class representing a basic stack ADT with push, pop, and peek methods.		
	2. Write a program that defines an ADT for a queue and implements it using a list in Python.		
	3. Create a Python class for a basic linked list ADT with methods to insert, delete, and traverse the list.		
	4. Develop a program that demonstrates the use of a Python list as an array ADT.		
	5. Implement a Python ADT for a set, including methods for union, intersection, and difference.		
13 Introduction to Data Structures	1. Write a Python program to create and manipulate a 2D array.		
	2. Implement a basic Python class for a singly linked list and add methods for insertion and deletion.		
	3. Create a program to perform arithmetic operations on polynomials using arrays.		
	4. Develop a Python class representing a doubly linked list with methods for insertion and deletion.		
	5. Write a program to create a circular linked list and perform basic operations on it.		
14 Stacks	1. Implement a Python program that checks the balanced parentheses in an expression using a stack.		
	2. Write a program to convert an infix expression to a postfix expression and evaluate it using a stack.		
	3. Create a Python class for a stack ADT and implement it using a linked list.		
14 Stacks	4. Develop a program that simulates a browser's forward and backward navigation using two stacks.		
	5. Implement a Python program to reverse a string using a stack.		
15 Queues	1. Write a Python program to implement a queue using two stacks.		
	2. Implement a program to simulate a printing queue with priority.		
	3. Create a Python class for a queue ADT and implement it using a linked list.		
	4. Develop a program that simulates a supermarket queue with customers entering and leaving.		
	5. Write a Python program to implement a priority queue for tasks with different priorities.		
16 Binary Trees	1. Implement a Python class for a binary tree and perform a depth-first traversal (preorder, inorder, postorder).		
	2. Write a program to find the height of a binary tree.		
	3. Create a Python function to check if a binary tree is a binary search tree.		
	4. Develop a program that performs a level-order traversal of a binary tree.		
	5. Implement a Python program to find the lowest common ancestor of two nodes in a binary tree.		
17 Binary Trees (Continued)	1. Write a Python program to create a binary search tree from a sorted list of values.		
	2. Implement a program that checks if a binary tree is balanced.		
	3. Create a Python function to serialize and deserialize a binary tree.		
	4. Develop a program to find the maximum value in a binary tree.		
	5. Write a Python program to find the diameter of a binary tree.		
18 Binary Search Trees	1. Implement a Python program to search for a value in a binary search tree.		
	2. Write a program to insert a value into a binary search tree.		
	3. Create a Python function to delete a node from a binary search tree.		
	4. Develop a program that finds the minimum and maximum values in a binary search tree.		
	5. Implement a Python program to check if two binary search trees are identical.		
19 Hash Tables	1. Write a Python program to implement a basic hash table with key-value pairs.		
	2. Implement a program that handles collisions in a hash table using chaining.		
	3. Create a Python function to hash a string and store it in a hash table.		
	4. Develop a program that searches for a key in a hash table and returns the corresponding value.		
	5. Write a Python program to remove a key-value pair from a hash table.		
20 Se	rithm to find an element in a list.		
	form a binary search on a sorted list.		
	arch for an element in a rotated sorted array.		
	he square root of a number using a binary search approach.		

DS Programming Questions

Day	Questions	Understood	Implem
1 Intro to Programming	1. Write a Python program to calculate the factorial of a number.		
	2. Implement a Python program to find the sum of even numbers in a list.		
	3. Create a Python function to reverse a given string.		
	4. Write a program to check if a number is prime.		
	5. Develop a Python program to solve a basic math problem, e.g., finding the area of a rectangle.		
2 Variables, Data Types	1. Write a Python program to convert temperature from Fahrenheit to Celsius.		
	2. Implement a program to calculate the area of a circle.		
	3. Create a Python function to check if a given number is even or odd.		
	4. Write a program to find the square root of a number.		
	5. Develop a Python program to calculate the simple interest.		
3 Control Flow	1. Write a Python program to check if a year is a leap year.		
	2. Implement a program to find the maximum of three numbers.		
	3. Create a Python function to calculate the sum of natural numbers up to N.		
	4. Write a program to print the Fibonacci sequence up to N terms.		
	5. Develop a Python program to check if a number is a palindrome.		
4 Collections	1. Write a Python program to count the number of vowels in a string.		
	2. Implement a program to find the intersection of two lists.		
	3. Create a Python function to reverse a given list.		
	4. Write a program to count the frequency of elements in a list.		
	5. Develop a Python program to remove duplicates from a list.		
5 Functions	1. Implement a Python program to calculate the power of a number using a recursive function.		
	2. Write a program to find the LCM of two numbers using a function.		
	3. Create a Python function to generate a random password of a given length.		
	4. Write a program to find the GCD of two numbers using a function.		
	5. Develop a Python program to check if a string is a palindrome using a function.		
6 Functions	1. Implement a Python program to calculate the exponential of a number using a user-defined function.		
	2. Write a program to find the mean and median of a list of numbers using functions.		
	3. Create a Python function to check if a given string is a valid palindrome (ignoring spaces and punctuation).		
	4. Write a program to generate a Fibonacci sequence using a recursive function.		
	5. Develop a Python program to find the greatest common divisor (GCD) of two numbers using a recursive		
7 Scope of Variables and Recursion	1. Write a Python program to demonstrate the scope of variables in different functions.		
	2. Implement a program to calculate the sum of natural numbers using recursion.		
	3. Create a recursive function to find the factorial of a number.		
	4. Write a program to solve the Tower of Hanoi puzzle using recursion.		
	5. Develop a Python program to compute the nth Fibonacci number using recursion.		
8 Object-Oriented Programming (OOP) Basics	1. Implement a Python class representing a simple bank account with deposit and withdrawal methods.		
	2. Write a Python program to create an instance of a car object with attributes like make, model, and year.		
	3. Create a Python class for a basic calculator that can add, subtract, multiply, and divide.		
	4. Write a program to create a Python class representing a book with attributes like title and author.		
	5. Develop a Python program to demonstrate the concept of encapsulation using a class.		
9 OOP Concepts (Constructors, Class)	1. Implement a Python class representing a student with a constructor and methods for setting and getting student details.		
	2. Write a Python program to create a class diagram for a simple online shopping system.		
	3. Create a Python class for a basic shape with constructor and methods to calculate area and perimeter.		
	4. Develop a Python program that demonstrates the concept of inheritance with multiple classes.		
	5. Write a program to create a Python class representing a basic employee with attributes like name, id, and		
10 More OOP (Statics, Relationship, Inheritance, Abstract Classes)	1. Implement a Python class with a static variable to keep track of the number of instances created.		
	2. Write a Python program to create a class relationship diagram for a library system.		
	3. Create a Python class representing a geometric shape with methods for calculating area and perimeter.		
	4. Develop a Python program to demonstrate single and multiple inheritance with classes.		
	5. Write a program to create an abstract class representing a vehicle with abstract methods for moving and		
11 Exception Handling	1. Write a Python program that demonstrates the use of a try-except block to handle a division by zero error.		
	2. Implement a program that raises a custom exception when a user tries to withdraw more money from an ATM than the account balance.		
	3. Create a Python function that takes a list as input and handles an "Index out of Range" exception.		
	4. Develop a program that uses a try-except block to handle a FileNotFoundError when reading data from a file.		
	5. Write a Python program that demonstrates the use of multiple except blocks for different types of exceptions.		
12 Abstract Data Types (ADT)	1. Implement a Python class representing a basic stack ADT with push, pop, and peek methods.		
	2. Write a program that defines an ADT for a queue and implements it using a list in Python.		
	3. Create a Python class for a basic linked list ADT with methods to insert, delete, and traverse the list.		
	4. Develop a program that demonstrates the use of a Python list as an array ADT.		
	5. Implement a Python ADT for a set, including methods for union, intersection, and difference.		
13 Introduction to Data Structures	1. Write a Python program to create and manipulate a 2D array.		
	2. Implement a basic Python class for a singly linked list and add methods for insertion and deletion.		
	3. Create a program to perform arithmetic operations on polynomials using arrays.		
	4. Develop a Python class representing a doubly linked list with methods for insertion and deletion.		
	5. Write a program to create a circular linked list and perform basic operations on it.		
14 Stacks	1. Implement a Python program that checks the balanced parentheses in an expression using a stack.		
	2. Write a program to convert an infix expression to a postfix expression and evaluate it using a stack.		
	3. Create a Python class for a stack ADT and implement it using a linked list.		
14 Stacks	4. Develop a program that simulates a browser's forward and backward navigation using two stacks.		
	5. Implement a Python program to reverse a string using a stack.		
15 Queues	1. Write a Python program to implement a queue using two stacks.		
	2. Implement a program to simulate a printing queue with priority.		
	3. Create a Python class for a queue ADT and implement it using a linked list.		
	4. Develop a program that simulates a supermarket queue with customers entering and leaving.		
	5. Write a Python program to implement a priority queue for tasks with different priorities.		
16 Binary Trees	1. Implement a Python class for a binary tree and perform a depth-first traversal (preorder, inorder, postorder).		
	2. Write a program to find the height of a binary tree.		
	3. Create a Python function to check if a binary tree is a binary search tree.		
	4. Develop a program that performs a level-order traversal of a binary tree.		
	5. Implement a Python program to find the lowest common ancestor of two nodes in a binary tree.		
17 Binary Trees (Continued)	1. Write a Python program to create a binary search tree from a sorted list of values.		
	2. Implement a program that checks if a binary tree is balanced.		
	3. Create a Python function to serialize and deserialize a binary tree.		
	4. Develop a program to find the maximum value in a binary tree.		
	5. Write a Python program to find the diameter of a binary tree.		
18 Binary Search Trees	1. Implement a Python program to search for a value in a binary search tree.		
	2. Write a program to insert a value into a binary search tree.		
	3. Create a Python function to delete a node from a binary search tree.		
	4. Develop a program that finds the minimum and maximum values in a binary search tree.		
	5. Implement a Python program to check if two binary search trees are identical.		
19 Hash Tables	1. Write a Python program to implement a basic hash table with key-value pairs.		
	2. Implement a program that handles collisions in a hash table using chaining.		
	3. Create a Python function to hash a string and store it in a hash table.		
	4. Develop a program that searches for a key in a hash table and returns the corresponding value.		
	5. Write a Python program to remove a key-value pair from a hash table.		
20 Se	rithm to find an element in a list.		
	form a binary search on a sorted list.		
	arch for an element in a rotated sorted array.		
	he square root of a number using a binary search approach.		

DS Programming Questions

Day	Questions	Understood	Implem
1 Intro to Programming	1. Write a Python program to calculate the factorial of a number.		
	2. Implement a Python program to find the sum of even numbers in a list.		
	3. Create a Python function to reverse a given string.		
	4. Write a program to check if a number is prime.		
	5. Develop a Python program to solve a basic math problem, e.g., finding the area of a rectangle.		
2 Variables, Data Types	1. Write a Python program to convert temperature from Fahrenheit to Celsius.		
	2. Implement a program to calculate the area of a circle.		
	3. Create a Python function to check if a given number is even or odd.		
	4. Write a program to find the square root of a number.		
	5. Develop a Python program to calculate the simple interest.		
3 Control Flow	1. Write a Python program to check if a year is a leap year.		
	2. Implement a program to find the maximum of three numbers.		
	3. Create a Python function to calculate the sum of natural numbers up to N.		
	4. Write a program to print the Fibonacci sequence up to N terms.		
	5. Develop a Python program to check if a number is a palindrome.		
4 Collections	1. Write a Python program to count the number of vowels in a string.		
	2. Implement a program to find the intersection of two lists.		
	3. Create a Python function to reverse a given list.		
	4. Write a program to count the frequency of elements in a list.		
	5. Develop a Python program to remove duplicates from a list.		
5 Functions	1. Implement a Python program to calculate the power of a number using a recursive function.		
	2. Write a program to find the LCM of two numbers using a function.		
	3. Create a Python function to generate a random password of a given length.		
	4. Write a program to find the GCD of two numbers using a function.		
	5. Develop a Python program to check if a string is a palindrome using a function.		
6 Functions	1. Implement a Python program to calculate the exponential of a number using a user-defined function.		
	2. Write a program to find the mean and median of a list of numbers using functions.		
	3. Create a Python function to check if a given string is a valid palindrome (ignoring spaces and punctuation).		
	4. Write a program to generate a Fibonacci sequence using a recursive function.		
	5. Develop a Python program to find the greatest common divisor (GCD) of two numbers using a recursive		
7 Scope of Variables and Recursion	1. Write a Python program to demonstrate the scope of variables in different functions.		
	2. Implement a program to calculate the sum of natural numbers using recursion.		
	3. Create a recursive function to find the factorial of a number.		
	4. Write a program to solve the Tower of Hanoi puzzle using recursion.		
	5. Develop a Python program to compute the nth Fibonacci number using recursion.		
8 Object-Oriented Programming (OOP) Basics	1. Implement a Python class representing a simple bank account with deposit and withdrawal methods.		
	2. Write a Python program to create an instance of a car object with attributes like make, model, and year.		
	3. Create a Python class for a basic calculator that can add, subtract, multiply, and divide.		
	4. Write a program to create a Python class representing a book with attributes like title and author.		
	5. Develop a Python program to demonstrate the concept of encapsulation using a class.		
9 OOP Concepts (Constructors, Class)	1. Implement a Python class representing a student with a constructor and methods for setting and getting student details.		
	2. Write a Python program to create a class diagram for a simple online shopping system.		
	3. Create a Python class for a basic shape with constructor and methods to calculate area and perimeter.		
	4. Develop a Python program that demonstrates the concept of inheritance with multiple classes.		
	5. Write a program to create a Python class representing a basic employee with attributes like name, id, and		
10 More OOP (Statics, Relationship, Inheritance, Abstract Classes)	1. Implement a Python class with a static variable to keep track of the number of instances created.		
	2. Write a Python program to create a class relationship diagram for a library system.		
	3. Create a Python class representing a geometric shape with methods for calculating area and perimeter.		
	4. Develop a Python program to demonstrate single and multiple inheritance with classes.		
	5. Write a program to create an abstract class representing a vehicle with abstract methods for moving and		
11 Exception Handling	1. Write a Python program that demonstrates the use of a try-except block to handle a division by zero error.		
	2. Implement a program that raises a custom exception when a user tries to withdraw more money from an ATM than the account balance.		
	3. Create a Python function that takes a list as input and handles an "Index out of Range" exception.		
	4. Develop a program that uses a try-except block to handle a FileNotFoundError when reading data from a file.		
	5. Write a Python program that demonstrates the use of multiple except blocks for different types of exceptions.		
12 Abstract Data Types (ADT)	1. Implement a Python class representing a basic stack ADT with push, pop, and peek methods.		
	2. Write a program that defines an ADT for a queue and implements it using a list in Python.		
	3. Create a Python class for a basic linked list ADT with methods to insert, delete, and traverse the list.		
	4. Develop a program that demonstrates the use of a Python list as an array ADT.		
	5. Implement a Python ADT for a set, including methods for union, intersection, and difference.		
13 Introduction to Data Structures	1. Write a Python program to create and manipulate a 2D array.		
	2. Implement a basic Python class for a singly linked list and add methods for insertion and deletion.		
	3. Create a program to perform arithmetic operations on polynomials using arrays.		
	4. Develop a Python class representing a doubly linked list with methods for insertion and deletion.		
	5. Write a program to create a circular linked list and perform basic operations on it.		
14 Stacks	1. Implement a Python program that checks the balanced parentheses in an expression using a stack.		
	2. Write a program to convert an infix expression to a postfix expression and evaluate it using a stack.		
	3. Create a Python class for a stack ADT and implement it using a linked list.		
14 Stacks	4. Develop a program that simulates a browser's forward and backward navigation using two stacks.		
	5. Implement a Python program to reverse a string using a stack.		
15 Queues	1. Write a Python program to implement a queue using two stacks.		
	2. Implement a program to simulate a printing queue with priority.		
	3. Create a Python class for a queue ADT and implement it using a linked list.		
	4. Develop a program that simulates a supermarket queue with customers entering and leaving.		
	5. Write a Python program to implement a priority queue for tasks with different priorities.		
16 Binary Trees	1. Implement a Python class for a binary tree and perform a depth-first traversal (preorder, inorder, postorder).		
	2. Write a program to find the height of a binary tree.		
	3. Create a Python function to check if a binary tree is a binary search tree.		
	4. Develop a program that performs a level-order traversal of a binary tree.		
	5. Implement a Python program to find the lowest common ancestor of two nodes in a binary tree.		
17 Binary Trees (Continued)	1. Write a Python program to create a binary search tree from a sorted list of values.		
	2. Implement a program that checks if a binary tree is balanced.		
	3. Create a Python function to serialize and deserialize a binary tree.		
	4. Develop a program to find the maximum value in a binary tree.		
	5. Write a Python program to find the diameter of a binary tree.		
18 Binary Search Trees	1. Implement a Python program to search for a value in a binary search tree.		
	2. Write a program to insert a value into a binary search tree.		
	3. Create a Python function to delete a node from a binary search tree.		
	4. Develop a program that finds the minimum and maximum values in a binary search tree.		
	5. Implement a Python program to check if two binary search trees are identical.		
19 Hash Tables	1. Write a Python program to implement a basic hash table with key-value pairs.		
	2. Implement a program that handles collisions in a hash table using chaining.		
	3. Create a Python function to hash a string and store it in a hash table.		
	4. Develop a program that searches for a key in a hash table and returns the corresponding value.		
	5. Write a Python program to remove a key-value pair from a hash table.		
20 Se	rithm to find an element in a list.		
	form a binary search on a sorted list.		
	arch for an element in a rotated sorted array.		
	he square root of a number using a binary search approach.		