Coding related hr qs

**1. Problem-Solving Questions**

**Q: Write a function to check if a string is a palindrome.**

**A:**

python

Copy code

def is\_palindrome(s):

return s == s[::-1]

# Example usage:

print(is\_palindrome("madam")) # True

print(is\_palindrome("hello")) # False

**2. Data Structures and Algorithms**

**Q: What is the difference between an array and a linked list? Can you write code to implement a linked list?**

**A:**

* **Array**: Fixed size, contiguous memory allocation, fast access by index.
* **Linked List**: Dynamic size, non-contiguous memory allocation, slower access since you need to traverse the list.

**Linked List Implementation in Python:**

python

Copy code

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.next = None

class LinkedList:

def \_\_init\_\_(self):

self.head = None

def append(self, data):

new\_node = Node(data)

if not self.head:

self.head = new\_node

return

last\_node = self.head

while last\_node.next:

last\_node = last\_node.next

last\_node.next = new\_node

def print\_list(self):

temp = self.head

while temp:

print(temp.data, end=" -> ")

temp = temp.next

print("None")

# Example usage:

ll = LinkedList()

ll.append(1)

ll.append(2)

ll.append(3)

ll.print\_list() # 1 -> 2 -> 3 -> None

**3. Sorting and Searching Algorithms**

**Q: Write a function to sort an array using Bubble Sort.**

**A:**

python

Copy code

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]

# Example usage:

arr = [64, 34, 25, 12, 22, 11, 90]

bubble\_sort(arr)

print("Sorted array:", arr) # [11, 12, 22, 25, 34, 64, 90]

**4. Time and Space Complexity**

**Q: What is the time complexity of Bubble Sort?**

**A:**

* **Time Complexity**: O(n^2), where n is the number of elements in the array.
* **Space Complexity**: O(1) because Bubble Sort is an in-place sorting algorithm.

**5. Object-Oriented Programming (OOP)**

**Q: Can you explain the four pillars of OOP (Encapsulation, Abstraction, Inheritance, and Polymorphism)?**

**A:**

* **Encapsulation**: Bundling data (attributes) and methods (functions) that operate on the data into a single unit called a class. It also refers to restricting access to some of the object's components to protect the object's state.
* **Abstraction**: Hiding the complexity and only exposing essential features of an object. For example, using abstract classes or interfaces to define methods without implementation.
* **Inheritance**: A mechanism that allows one class to inherit attributes and methods from another class, promoting code reuse. For example, a Dog class can inherit from an Animal class.
* **Polymorphism**: The ability to use a single interface to represent different underlying forms. For example, a method with the same name can have different implementations for different types of objects.

**6. Debugging**

**Q: Can you spot the error in this code?**

python

Copy code

def sum\_elements(lst):

total = 0

for i in lst:

total += i

return total

# Mistake: You forgot to call the function.

# Corrected version:

print(sum\_elements([1, 2, 3, 4])) # Output: 10

**7. Code Optimization**

**Q: Can you optimize this code to find the maximum element in an array?**

**Original Code:**

python

Copy code

def find\_max(arr):

max\_val = arr[0]

for i in arr:

if i > max\_val:

max\_val = i

return max\_val

**Optimized Code:**

python

Copy code

# The original code is already optimal with O(n) time complexity.

# The optimization would be in terms of readability.

def find\_max(arr):

return max(arr) # Using built-in function

# Example usage:

arr = [1, 2, 3, 4]

print(find\_max(arr)) # 4

**8. Real-Life Scenario Coding**

**Q: Write a program that takes an array and removes duplicate values.**

**A:**

python

Copy code

def remove\_duplicates(arr):

return list(set(arr))

# Example usage:

arr = [1, 2, 2, 3, 4, 4]

print(remove\_duplicates(arr)) # [1, 2, 3, 4]

**9. Code Explanation**

**Q: Explain the following code snippet:**

python

Copy code

def fibonacci(n):

if n <= 1:

return n

return fibonacci(n-1) + fibonacci(n-2)

**A:**

* This code is a recursive implementation of the Fibonacci sequence.
* The Fibonacci sequence starts with 0 and 1, and each subsequent number is the sum of the previous two.
* For example, fibonacci(4) would return 3 because the sequence is 0, 1, 1, 2, 3.
* The function works by recursively calling itself with n-1 and n-2 until it reaches the base case (n <= 1).

**10. Language-Specific Questions**

**Q: What is the difference between == and is in Python?**

**A:**

* ==: Compares the values of two objects.
* is: Compares the identity (memory location) of two objects.

Example:

python

Copy code

a = [1, 2, 3]

b = [1, 2, 3]

print(a == b) # True, since values are equal

print(a is b) # False, since they are different objects in memory