PRACTICE SHEET

What you will Learn:

Shallow vs Deep Copy & Memory Behavior, List & Dictionary Comprehensions, Identity vs Equality, Copy Modules(copy() vs deepcopy()), Tuples vs Lists – Memory & Performance, Linked List Basics & Duplicate Removal

[EASY] Q1) What will be the output of the following code?

```
x = [10, 20, [30, 40]]

y = x.copy()

y[2].append(50)

print(x)
```

Options:
a) [10, 20, [30, 40]]
b) [10, 20, [30, 40, 50]]
c) [10, 20, [30, 40], 50]
d) Error

[EASY] Q2) Which of the following comprehensions creates a dictionary where keys are integers from 1 to 5 and values are their cubes?

Options:
a) {x: x**3 for x in range(1, 6)}
b) {x**3: x for x in range(1, 6)}
c) {x: x*x for x in range(1, 6)}
d) {x: x*3 for x in range(1, 6)}

[EASY] Q3**)** Consider the following snippet:

```
a = [1, 2, 3]

b = a

c = a.copy()
```

```
print(a is b, a == c)
```

What is the output?

Options:
a) True True
b) True False
c) False True
d) False False

[EASY] Q4) What does del do in the following context?

```
x = [1, 2, 3]

y = x

del x

print(y)
```

Options:
a) Error
b) Prints [1, 2, 3]
c) Prints None
d) Deletes y too

[EASY] Q5) Which memory function is used to determine whether two objects are exactly the same in memory?

Options:

a) ==

b) copy()

c) id()

d) deepcopy()

[MEDIUM] Q6) You have a production line system where each line stores status in nested lists. Prove that shallow copy causes data leakage across systems:

```
Line1 = [[1, ‘running’], [2, ‘halted’]]

Line2 = copy.copy(Line1)

Line2[0][1] = ‘stopped’

Print(Line1)
```

Task: Explain why line1 gets changed and fix it using deepcopy.

[MEDIUM] Q7) Given a list of voltage readings across multiple resistors:

```
Voltages = [12, 24, 36, 48]
```

Use a list comprehension to:

Calculate power drop across each resistor using formula P = V^2 / R.

Assume resistance R = 12 ohms.

Return list of all power drops where result is more than 50W.

[MEDIUM] Q8) Given logs of packet IDs received multiple times:

```
Logs = [101, 105, 106, 101, 105, 107, 108]
```

Write a set comprehension to extract all duplicate packet IDs (i.e., IDs received more than once).

[HARD] Q9) You’re developing an IoT system where 100 sensors report temperature data every minute. Some sensors report duplicate readings. You want to:

Extract all unique faulty readings that occur more than once.

Identify if two sensor data objects are pointing to the same memory block, due to a faulty data forwarding mechanism.

Tasks:

Use set comprehension to extract all values occurring more than once from a list.

Verify whether two faulty reading variables are referring to the same memory object using is and id().

```
Sensor_data = [23.4, 24.1, 23.4, 25.5, 24.1, 24.1]
```

[MEDIUM] Q10) You are simulating cache in a distributed database. The same object is referenced by multiple cache nodes. Modifying it at one node unintentionally changes it in another.

```
Cache_node_1 = {“query”: “SELECT * FROM users”, “count”: 5}

Cache_node_2 = cache_node_1
```

Task:

Modify cache_node_2[“count”] = 10 and show it also changes cache_node_1

Use copy() and deepcopy() to prevent this.

[EASY] Q11) Which is True about memory for tuple vs list?

A. Tuples consume more memory than lists

B. Tuples are faster and consume less memory than lists

C. Both are equal

D. Tuples are slower due to immutability

[EASY] Q12) You are an embedded engineer storing device configurations.

Compare memory of the following:

```
import sys

print(sys.getsizeof([1, 2, 3, 4]))

print(sys.getsizeof((1, 2, 3, 4)))
```

Write code and explain which is lighter and why you’d use it in low-memory environments.

[MEDIUM] Q13) Cube is given an integer n. She wants to know how many ordered pairs of positive integers (a,b) there are such that $a=n-b$. Since Cube is not very good at math, please help her!

Input: The first line contains an integer t - the number of test cases. The only line of each test case contains an integer n.

Output: For each test case, output the number of ordered pairs (a,b)(a,b) on a new line.

Example:

Input

3

2

4

6

Output

1

3

5

Note: In the first test case, the only ordered pair that works is (a,b)=(1,1). In the second test case, the three ordered pairs of (a,b) that work are (3,1),(2,2),(1,3).

[HARD] Q14) In order to test his patients' intelligence, Dr. TC created the following test.

First, he creates a binary string s having n characters. Then, he creates n binary strings $a_1,a_2,\dots,a_n$. It is known that a_i is created by first copying s, then flipping the i'th character (1 becomes 0 and vice versa). After creating all n strings, he arranges them into a grid where the i'th row is a_i.

For example,

- If s=101, a=[001,111,100]
- If s=0000, a=[1000,0100,0010,0001].

The patient needs to count the number of 1s written on the board in less than a second. Can you pass the test?

A binary string is a string that only consists of characters 1 and 0.

Input : The first line of the input consists of a single integer t - the number of test cases. The first line of each test case contains a single integer n - the length of the binary string s. The second line of each test case contains a single binary string ss of size n.

Output : For each test case, output a single integer, the number of 1s on the board.

Example:

Input

5

3

101

1

1

5

00000

2

11

3

010

Output

5

0

5

2

4

Note: For the second example, the only string written on the board will be the string 0; therefore, the answer is 0. In the third example, the following strings will be written on the board: [10000,01000,00100,00010,00001] so there are five 1s written on the board.

[HARD] Q15) **Linked List Overview:**
A linked list is a linear data structure where elements are stored in nodes. Each node contains two parts:

1. **Data**: The value of the node.
2. **Next**: A reference (or pointer) to the next node in the sequence.

In Python, a linked list can be implemented using a class for the nodes. For example:

```
class Node:
  def __init__(self, data):
    self.data = data  # Stores the value
    self.next = None   # Points to the next node
```

The linked list starts at the head node, and the last node's next points to None.

Task: Given the head of a **sorted** linked list, your task is to delete all duplicate values such that each value appears only once. Return the modified linked list.

Hints:

After reading the array (to simulate a list), use a **list comprehension** to remove consecutive duplicates.

Then **shallow-copy** the result, modify one element, and check if the original changes.

Demonstrates both **comprehension** and **mutability/leak**.

Input: The first line contains an integer N — the length of the linked list. The second line contains N space-separated integers representing the values of the linked list nodes starting from the head.

Output: Return the head of the modified linked list after removing duplicates.

Examples:

Input:

5

1 1 6 8 8

Output:

```
1 6 8
```

Explanation:
The input linked list is 1 → 1 → 6 → 8 → 8. After removing duplicates, the list becomes 1 → 6 → 8.