Date: 07/02/2026

Duration: 1 hour 45

CONTINUOUS ASSESSMENT 1

II YEAR – SEMESTER 4

CSE23AE204 - PCP III

(B.Tech. E01,E02,E03,E05,E06)

SET D

Question 1: Problem Statement

You are part of an **enterprise risk analytics team**.

A forecasting engine models **hierarchical risk propagation**, where each level derives its values from the **two adjacent values** in the level above it.

To allow automated validation, the engine exports the matrix as:

- A **row-wise hierarchical layout**
- Each row stored as a **string**
- Values computed using previously generated data

Your task is to generate this hierarchical coefficient matrix for a given integer N.

Input Format

- A single integer N representing the number of levels in the hierarchy.

Output Format

- Return an **array/list of strings**
- Each string represents one row of the matrix

Example 1

Input

5

Output

```
[
 "1",
 "1 1",
 "1 2 1",
 "1 3 3 1",
 "1 4 6 4 1"
]
```

Question 2: Problem Statement

You are given a string **s** consisting of uppercase English letters.

Your task is to compress the string using a **custom adaptive run-length encoding rule** based on the **frequency comparison with the previous group**.

1. Group **consecutive identical characters**.
2. Let:
   - currFreq = frequency of the current character group
   - prevFreq = frequency of the previous character group
3. Encoding rule:
   - **First group** → encode as: <character><frequency>
   - For every next group:
     - If currFreq < prevFreq → encode as <character><frequency>
     - Else → encode as <frequency><character>
4. If frequency is 1, **omit the number**.
5. Maintain the original order of characters.

Input Format: s (string)

Output Format: compressed string

Example 1

Input

ABBBCCDAAAEE

Output

A3BC2D3AE2

Date: 07/02/2026

Duration: 1 hour 45

CONTINUOUS ASSESSMENT 1

II YEAR – SEMESTER 4

CSE23AE204 - PCP III

(B.Tech. E01,E02,E03,E05,E06)

SET D

Question 3: Problem Statement

You are developing a **fare calculation module** for a ride-sharing application. The system processes **one ride request at a time**, provided as a **space-separated string**. Each ride belongs to a specific category and follows different fare calculation rules.

The system must:

- Identify the ride type from input
- Calculate the total fare based on distance and ride category
- Use **inheritance and polymorphism** to support multiple ride types
- Assign a unique ride ID using a **static variable**
- Return the final fare amount

Fare Rules

Mini Ride

- A base fare is charged.
- An additional per-kilometer charge is applied based on distance.
- The total fare is the sum of base fare and distance charge.

Sedan Ride

- A higher base fare is charged compared to Mini.
- A higher per-kilometer charge is applied.
- The total fare is calculated accordingly.

SUV Ride

- The highest base fare is charged.
- The highest per-kilometer charge is applied.
- The total fare is calculated accordingly.

Input Format

A **single space-separated string**.

Mini Ride: MINI <RiderName> <DistanceInKm>

Sedan Ride: SEDAN <RiderName> <DistanceInKm>

SUV Ride: SUV <RiderName> <DistanceInKm>

Output Format

Return a **double value** representing the **total ride fare**.

Example 1

Input

{ "ride": "MINI Rahul 10" }

Output

150.0

Explanation

The base fare and per-kilometer charges applicable to a Mini ride are applied for 10 km.