1. Vowel Puzzle Checker

Scenario:
A word puzzle game accepts only words that:

- Start and end with a vowel
- Have exactly 2 vowels inside the word (excluding first and last)

Task:
Take a string input and check if it's a valid puzzle word.

Input: `"abide"`
Output: `"Valid"` (starts and ends with vowels, 2 vowels inside)

🔹 2. Airport Boarding Gate Simulation

Scenario:
Passengers are allowed to board in the order of their ticket numbers (integers). However, some passengers try to jump ahead.

Task:
Given an array of actual boarding order and expected order (sorted), check how many passengers are out of order.

Input: `[101, 103, 102, 104]`
Output: `1` (Only 103 and 102 are swapped)

🔹 3. Team Assignment Game

Scenario:
You are splitting `N` players into 2 teams such that:

- Total number of players in each team is equal
- The total skill score of both teams is as close as possible

Task:
Write a logic to divide players (represented by an array of skill scores) into two balanced teams.

Input: `[10, 20, 30, 40]`
Output: `Team A: [10, 40] | Team B: [20, 30]`

Bonus: Minimize skill score difference.

🔹 4. Student Rank List Generator

Scenario:
You have an array of marks. Your task is to:

- Sort the marks in descending order
- Print ranks (1st, 2nd, 3rd...)

Input: `[50, 80, 60]`
Output:

markdown
CopyEdit
`1. 80`
`2. 60`
`3. 50`

🔹 5. Palindrome Slot Machine

Scenario:
A slot machine gives you 3-digit numbers. You win if the number is a palindrome **and** the sum of digits is divisible by 3.

Input: `363`
Output: `"Jackpot!"`
(`363` is a palindrome, 3+6+3 = 12 → divisible by 3)

🔹 6. Bus Seat Allotment Simulator

Scenario:
A bus has 40 seats. Seats `1-10` are window seats.
A passenger input is an array of preferred seats. You must:

- Accept only available seats
- Prioritize window seats if available
- Reject if the seat is already booked

Input: `[3, 5, 12, 5, 11]`
Output:

nginx
CopyEdit

```
Seat 3 Booked
Seat 5 Booked
Seat 12 Booked
Seat 5 Already Booked
Seat 11 Booked
```

🔹 7. Custom Password Encoder

Scenario:
You are designing a password encoder where:

- Replace vowels with `@`
- Replace even digits with `*`
- Convert everything to uppercase

Input: `"Java1234"`
Output: `"J@V@1*3*"`

🔹 8. Exam Hall Seat Validator

Scenario:
You have a row of student IDs sitting in seats.
No two students with the same last digit of ID can sit adjacent.

Input: `[21, 34, 43, 52]`
Output: `"Invalid - 34 and 43 have same last digit"`

🔹 9. Pattern Lock Attempt Tracker

Scenario:
You allow a user 5 attempts to enter a 4-digit pattern.
If the correct pattern (`1234`) is entered within 5 tries, print success.
Else, lock the system.

Input: User enters patterns one by one.

🔹 10. Odd Digit Reverser

Scenario:
You're given a number. Reverse it, but only include the **odd digits**.

Input: `123456789`
Output: `97531`