Coding Exercises

Q1

Design an OOP model for planning trip fuel across multiple vehicle types.
Requirements:
- Provide a general vehicle type with encapsulated core data (private fields) and validated constructors (invalid → print an error; keep previous values).
- Create at least two specialized vehicle types that inherit from the general type and introduce one private field each affecting fuel usage, with validation.
- Define a fuel computation method in the general type; specialized types must override it with their own rule.
- In a mixed collection of vehicles, given a list of trip distances, compute total fuel per vehicle and print which vehicles cannot complete the route under their own constraints (you define the constraint per type).

Q2

Model shapes to compute total paintable area and cost.
Requirements:
- Provide a general shape type (concrete class) with an *area()* method that can be overridden.
- Implement at least three concrete shape types with encapsulated dimensions and validated constructors (invalid → print; keep previous).
- Use polymorphism with a mixed collection of shapes to compute total area (no type checks in client code).
- Apply tiered pricing: first 50 units at 1.50, next 100 at 1.25, remainder at 1.00; print total area and total cost to 2 decimals.

Q3

Given an array of integers *nums* sorted in ascending order, and an integer *target*, write a function to search *target* in *nums*.
- If *target* exists, return its index. Otherwise, return -1.
- The algorithm must run in **O(log n)** time complexity.

Examples:
- Input: nums = [-1,0,3,5,9,12], target = 9 → Output: 4
Explanation: 9 exists in nums and its index is 4.
- Input: nums = [-1,0,3,5,9,12], target = 2 → Output: -1
Explanation: 2 does not exist in nums, so return -1.