1. Design a Book class for a library system that's in desperate need of organization, with attributes title, author, and price—because even imaginary books need royalties. Implement a default constructor that sets the blandest values possible (think "Untitled," "Anonymous," and free), a parameterized constructor for when you want your book to have an actual identity, a copy constructor for duplicating your masterpiece, and a constructor with default arguments where you only need to provide the title, while author defaults to "Unknown" and price to 0.0 (clearly a bestseller). Test your creation by spawning book objects every way imaginable and proudly showing off their details like they're competing in a talent show.

2. Create a class called WeekDays that's here to help you never lose track of what day it is (because let's face it, it happens). This class has two private data members:

    - **Days** – A string array with the seven glorious days of the week (Sunday to Saturday).
    - **CurrentDay** – An integer variable to track which day you're currently living in.

    Now, implement the following constructors and member functions:
    - **Default Constructor** – Fills up the **Days** array with the usual suspects: Sunday, Monday, Tuesday… you get it.
    - **Parameterized Constructor** – Takes an integer for **CurrentDay**. If the value is greater than 6 (because there's no Day 8), do some math magic with modulus (%) to bring it back within range. For example, if the input is 8, 8 % 7 = 1, so **CurrentDay** becomes Monday. Oh, and don't forget to fill the **Days** array too.
    - **getCurrentDay** – Returns the name of the current day. (Think Days[CurrentDay].)
    - **getNextDay** – Returns the name of tomorrow, because who doesn't want to know what day's coming next?
    - **getPreviousDay** – Returns yesterday's name, just in case you're feeling nostalgic.
    - **getNthDayFromToday** – Takes an integer N and tells you what day it will be N days from now. For example, if today is Monday and N = 20, it'll tell you that it's Sunday in 20 days. Useful for planning vacations or just confusing yourself for fun!

3. Create a SmartDevice class to represent your high-tech home gadgets—because flipping switches manually is so last century. Give it attributes like device name, type (smart light, robo-vacuum, coffee overlord), and status (on or off). Build a parameterized constructor to bring these gadgets to life and a destructor that delivers an emotional farewell message when they meet their inevitable end. Test it by creating and deleting objects, and enjoy watching each device's dramatic exit as they "power down" for the last time.

4. Create a CricketPlayer class to represent the superstars (or struggling rookies) of the Pakistan cricket team, with attributes like name, jerseyNumber, and battingAverage—because every player needs something to brag about on Instagram. Use a parameterized constructor with the this keyword to avoid variable confusion, or worse, run-outs. Add a method improveAverage(runs) that magically boosts the batting average and returns the object for method chaining—because why stop at one record when you can break several? Create displayPlayerStats() to proudly show off the player's details, complete with virtual commentary like, "This batting average is more consistent than Karachi's weather!"
   To spice things up, add playMatch(runsScored) to update the batting average after each match, so players can experience the highs of centuries and the lows of golden ducks. Test it by creating players (Babar Azam, Muhammad Rizwan, Saim Ayub) like your own dream squad, making them score big, boosting averages as if they're on fire, and then showing off their stats like they just hit six sixes in an over and became national heroes overnight!

5. Create a FootballPlayer class to manage superstar footballers, with attributes like playerName, position, and goalCount—because every player deserves a legendary profile. Implement a default constructor that sets values like "Unknown Player," "Benchwarmer," and 0 goals. Add a parameterized constructor for customizing football icons and a copy constructor for cloning players (because every team could use two Messis). Include a constructor with default arguments, where only playerName is required, while position defaults to "Midfielder" and goalCount defaults to 10 (just to flex a little). Test the class by creating footballers, boosting their goal counts, and displaying their profiles like they're about to win the Ballon d'Or!