C++ Basics Practice Questions
September 29, 2025

General Guidelines:

1. Read the question thoroughly. Understand the scenario, requirements, and expected functionality before you begin coding.
2. Use the exact variable names, function names, and class names provided in the question.
   This ensures consistency and helps in automated or manual evaluation.
3. Follow all technical requirements as specified.
4. Follow MISRA C++ naming conventions.
   - Use descriptive and consistent variable names (e.g., componentIdentifier, not id)
   - Avoid abbreviations and reserved keywords
   - Maintain proper formatting and indentation
5. Ensure your code is modular, readable, and well-commented.
   - Break down logic into functions
   - Add comments to explain key parts of your code
6. Validate all user inputs.
   - Check for valid ranges, non-empty strings, and correct enum selections
7. Test your program with the sample data provided.
   - Ensure all functionalities work as expected
8. Submit only the required .cpp and .h files.
   - Make sure it compiles and runs without errors
9. Do not modify the structure of the question.
   - Stick to the design and flow as described
10. Assume suitable data whenever required.
11. If you are not aware of few functionalities then those implementations can be ignored.

Reference Book: C++ How to Program
Authors: Paul Deitel, Harvey Deitel

Refer the above book for Task 1, Task 2 and Task 3. Task 4 and Task 5 presented from next page onwards.

Task 1: 8.3

Task 2: 8.4

Task 3: 8.5

Task 4: **The Tortoise and the Hare — A Race Through Code**

Task 4: The Tortoise and the Hare — A Race Through Code

Scenario

In the whimsical world of Codeville, two legendary racers—the steady **Tortoise** and the unpredictable **Hare**—are about to compete in a 70-square race to win a golden basket of carrots and lettuce. The race course winds up a slippery mountain, and with each tick of the clock, the racers may advance, slip, or even collide. Your task is to simulate this race using C++ and pointer-based functions to update each animal's position.

Objective

Write a C++ program that simulates the race between the tortoise and the hare. The simulation should use random number generation to determine each animal's movement per second and display the race progress in real time.

Requirements

1. **Track Setup**
   - The race track has 70 squares.
   - Both animals start at square 1.
   - The finish line is at square 70.
2. **Movement Rules**
   - Each second (clock tick), generate a random integer i∈[1,10]i \in [1,10] for each animal.
   - **Tortoise**:
     - Fast plod (50%): i∈[1,5]i \in [1,5] → move +3 squares
     - Slip (20%): i∈[6,7]i \in [6,7] → move −6 squares
     - Slow plod (30%): i∈[8,10]i \in [8,10] → move +1 square
   - **Hare**:
     - Sleep (20%): i∈[1,2]i \in [1,2] → no move
     - Big hop (20%): i∈[3,4]i \in [3,4] → move +9 squares
     - Big slip (10%): i=5i = 5 → move −12 squares
     - Small hop (30%): i∈[6,8]i \in [6,8] → move +1 square
     - Small slip (20%): i∈[9,10]i \in [9,10] → move −2 squares

3. **Position Constraints**
    - If an animal slips below square 1, reset its position to square 1.
4. **Function Design**
    - Implement moveTortoise(int* pos) and moveHare(int* pos) using pointer-based pass-by-reference to update positions.
5. **Race Display**
    - Print a 70-character line per tick showing:
        - T for tortoise
        - H for hare
        - OUCH!!! if both land on the same square
        - All other positions should be blank
6. **Race Start Message**

    BANG !!!!!

    AND THEY'RE OFF !!!!!
7. **Race Termination**
    - The race ends when one or both animals reach or pass square 70.
    - Print the result:
        - If tortoise wins: TORTOISE WINS!!! YAY!!!
        - If hare wins: Hare wins. Yuch.
        - If both win on the same tick: It's a tie. (or favor the tortoise)

Tasks to Perform in main()

1. Initialize positions of tortoise and hare to 1.
2. Print the race start message.
3. Loop until one or both reach square 70:
    - Call movement functions
    - Clamp positions to ≥ 1
    - Print race track line
    - Check for winner and break loop
4. Print final result.

Sample Data (Random i values and resulting positions)

Tick	Tortoise i	Move	Tortoise Pos	Hare i	Move	Hare Pos
1	3	+3	4	2	Sleep	1
2	6	−6	1	5	−12	1 → OUCH!!!
3	9	+1	2	7	+1	2 → OUCH!!!
4	1	+3	5	10	−2	1