C++ Programming Practice Set

New Set - Variables, Data Types, Scopes & Operators

EASY QUESTIONS (1-3)

Question 1: Employee Salary Calculator

Write a C++ program that:

- Declares variables for employee ID (int), employee name (string), basic salary (double), and years of experience (int)
- Takes input from user for all values
- Calculates bonus based on experience: 5% for <2 years, 10% for 2-5 years, 15% for >5 years
- Calculates total salary (basic + bonus)
- Uses appropriate data types and displays formatted output

Sample Input:

```
Enter Employee ID: 1001
Enter Employee Name: John Smith
Enter Basic Salary: 50000
Enter Years of Experience: 3
```

Expected Output:

```
=== EMPLOYEE DETAILS ===
ID: 1001
Name: John Smith
Basic Salary: $50000.00
Experience: 3 years
Bonus Rate: 10%
Bonus Amount: $5000.00
Total Salary: $55000.00
```

Question 2: Temperature Converter

Write a C++ program that:

- Takes a temperature value and unit (C/F/K) as input
- Uses arithmetic operators to convert between Celsius, Fahrenheit, and Kelvin
- Displays the temperature in all three units
- Use appropriate variable types (float/double for precision)

Conversion Formulas:

- C to F: F = (C × 9/5) + 32
- C to K: K = C + 273.15
- F to C: C = (F - 32) × 5/9

Sample Input:

```
Enter temperature: 25
Enter unit (C/F/K): C
```

Expected Output:

```
Original: 25.00°C
Fahrenheit: 77.00°F
Kelvin: 298.15K
```

Question 3: Number Property Checker

Write a C++ program that:

- Takes an integer as input
- Uses logical operators to check multiple properties:
  - Is it even or odd?
  - Is it positive, negative, or zero?
  - Is it divisible by both 3 and 5?
  - Is it a single digit number?
- Display all results using boolean logic and conditional output

Sample Input:

```
Enter a number: 15
```

Expected Output:

```
Number Analysis for 15:
Even: No
Positive: Yes
Divisible by 3: Yes
Divisible by 5: Yes
Divisible by both 3 and 5: Yes
Single digit: No
```

MEDIUM QUESTIONS (4-5)

Question 4: Banking System with Local/Global Variables

Write a C++ program that simulates a simple banking system:

- Use a global variable `totalBankBalance` to track overall bank money
- Create functions:
  - `depositMoney(double amount)` - adds to both account and total bank balance
  - `withdrawMoney(double amount)` - deducts if sufficient balance exists

- `displayBalances()` - shows both account and total bank balance

- In main(), demonstrate how local account balance and global bank balance work differently
- Use static local variable inside functions to count total transactions

Sample Run:

```
Initial Account Balance: 1000
Initial Bank Balance: 50000

After deposit of 500:
Account Balance: 1500
Total Bank Balance: 50500
Total Transactions: 1

After withdrawal of 200:
Account Balance: 1300
Total Bank Balance: 50300
Total Transactions: 2
```

Question 5: Bit Manipulation Utility

Write a C++ program that:

- Takes an integer and a bit position (0-31) as input
- Provides a menu for bit operations:
  1. Set bit at given position (use OR with left shift)
  2. Clear bit at given position (use AND with NOT)
  3. Toggle bit at given position (use XOR)
  4. Check if bit is set (use AND)
  5. Count total number of set bits
  6. Find position of rightmost set bit
- Display the number in binary format before and after operations

Sample Menu:

```
Enter number: 10
Enter bit position (0-31): 2

Current number: 10 (Binary: 1010)

Bit Operations Menu:
1. Set bit
2. Clear bit
3. Toggle bit
4. Check bit
5. Count set bits
6. Find rightmost set bit
Enter choice: 1

After setting bit 2: 14 (Binary: 1110)
```

HARD QUESTION (6)

Question 6: Scientific Calculator with Expression Parser

Write a C++ program that implements a scientific calculator:

Requirements:

- Create variables with different scopes (global constants, local variables, static variables)
- Implement functions for:
  1. `power(int base, int exp)` - using bitwise operations for optimization when exp is power of 2
  2. `factorial(int n)` - with static variable to cache last calculated value
  3. `gcd(int a, int b)` - using modulus operator
  4. `isPrime(int n)` - using logical operators and efficient checking

Main Calculator Features:

- Parse and evaluate complex expressions like: `(5^3) + (factorial(4) & 15) - (gcd(12,8) << 2)`
- Handle operator precedence correctly
- Use bitwise operations where applicable for optimization
- Demonstrate variable scope by using:
  - Global constants for mathematical values (PI = 3.14159)
  - Local variables for temporary calculations
  - Static variables for caching/counting
  - Block scope variables within conditional statements

Advanced Expression Example:

```cpp
// Your program should handle expressions like:
result = (power(2,4) & 0x0F) | (factorial(3) << 1) ^ (gcd(15,10) + isPrime(7) * 5)
```

Expected Program Structure:

```
Scientific Calculator
====================

Available Operations:
1. Power calculation (optimized for powers of 2)
2. Factorial (with caching)
3. GCD calculation
4. Prime checking
5. Complex expression evaluation
6. Show calculation statistics

Enter your choice: 5

Enter expression components:
Base for power: 3
Exponent: 4
Number for factorial: 5
First number for GCD: 18
Second number for GCD: 12
Number to check prime: 17

Evaluating: (3^4 & 31) | (5! >> 2) ^ (gcd(18,12) * isPrime(17))

Step-by-step calculation:
3^4 = 81
81 & 31 = 17
5! = 120 (cached result)
120 >> 2 = 30
gcd(18,12) = 6
isPrime(17) = 1 (true)
6 * 1 = 6
Final: 17 | 30 ^ 6 = 17 | 24 = 25

Result: 25 (Binary: 11001, Hex: 0x19)

Statistics:
Total calculations performed: 15
Cache hits: 1
Prime checks: 1
```