C++ Programming Practice Set 3

Control Flow Statements & Modular Programming with Functions

EASY QUESTIONS (1-3)

Question 1: Dice Game Simulator

Write a C++ program that:

- Simulates rolling two dice using random numbers (1-6)
- Uses if-else statements to determine winning conditions:
  - Snake Eyes: Both dice show 1 (lose all points)
  - Lucky Seven: Sum equals 7 (double points)
  - Double: Both dice same (bonus 10 points)
  - Normal: Add sum to score
- Uses a for loop to play multiple rounds
- Tracks and displays running score

Sample Run:

```
===== DICE GAME =====
Enter number of rounds: 5

Round 1: Rolling dice...
Dice 1: 4, Dice 2: 3
Sum: 7 - LUCKY SEVEN! Double points!
Score: 14

Round 2: Rolling dice...
Dice 1: 2, Dice 2: 2
Sum: 4 - DOUBLE! Bonus points!
Score: 28 (14 + 4 + 10 bonus)

Round 3: Rolling dice...
Dice 1: 1, Dice 2: 1
Sum: 2 - SNAKE EYES! All points lost!
Score: 0

Final Score: 0
```

Question 2: ATM PIN Validation System

Write a C++ program that:

- Uses a while loop to allow maximum 3 PIN attempts
- Takes 4-digit PIN input and validates against stored PIN (1234)
- Uses nested if statements to check each digit position
- Displays remaining attempts after each failed try

- Locks account after 3 failed attempts
- Uses do-while for transaction menu after successful login

Sample Interaction:

```
===== ATM SYSTEM =====
Enter 4-digit PIN: 1235
Incorrect PIN. 2 attempts remaining.

Enter 4-digit PIN: 4321
Incorrect PIN. 1 attempt remaining.

Enter 4-digit PIN: 1234
PIN Accepted. Welcome!

===== TRANSACTION MENU =====
1. Check Balance
2. Withdraw Money
3. Deposit Money
4. Exit

Enter choice: 1
Current Balance: $2,500.00

Continue? (Y/N): Y
```

Question 3: Library Book Return Calculator

Write a C++ program that:

- Takes book type (Fiction/Non-Fiction/Reference) using switch-case
- Calculates different borrowing periods: Fiction(14 days), Non-Fiction(21 days), Reference(7 days)
- Uses nested loops to calculate overdue fine:
  - Days 1-7 overdue: $0.50 per day
  - Days 8-14 overdue: $1.00 per day
  - Days 15+ overdue: $2.00 per day
- Displays detailed fine breakdown

Sample Input:

```
Enter book type:
1. Fiction (14 days)
2. Non-Fiction (21 days)
3. Reference (7 days)
Choice: 1

Enter days since borrowed: 20
```

Expected Output:

```
Book Type: Fiction
Allowed period: 14 days
Days borrowed: 20 days
Overdue by: 6 days

Fine Calculation:
Days 1-6 overdue: 6 × $0.50 = $3.00
Total Fine: $3.00
Please return the book and pay fine at counter.
```

MEDIUM QUESTIONS (4-5)

Question 4: Weather Data Analysis Functions

Write a C++ program with functions to analyze daily weather data:

- `void inputWeatherData(int days)` - inputs temperature and rainfall for multiple days
- `double calculateAverage(double data[], int size)` - calculates average using loops
- `int findExtremeDay(double data[], int size, bool findMax)` - finds hottest/coldest day
- `void classifyWeather()` - categorizes days as Hot(>30°C), Mild(15-30°C), Cold(<15°C)
- `void generateForecast()` - predicts next day weather using trend analysis
- Use arrays to store data and nested loops for analysis

Expected Program Flow:

```
===== WEATHER ANALYSIS SYSTEM =====
Enter number of days to analyze: 7

Day 1: Temperature(°C): 25, Rainfall(mm): 5
Day 2: Temperature(°C): 28, Rainfall(mm): 0
Day 3: Temperature(°C): 22, Rainfall(mm): 12
...
Day 7: Temperature(°C): 30, Rainfall(mm): 8

===== ANALYSIS RESULTS =====
Average Temperature: 26.4°C
Hottest Day: Day 7 (30°C)
Coldest Day: Day 3 (22°C)
Total Rainfall: 35mm

Weather Classification:
Hot days: 2 (Days 6, 7)
Mild days: 5 (Days 1, 2, 3, 4, 5)
Cold days: 0

Forecast for Day 8:
Temperature trend: Rising (+2°C over last 3 days)
Predicted temperature: 32°C
Rain probability: 40% (based on recent pattern)
```

Question 5: Gaming Tournament Bracket System

Write a C++ program with functions to manage a tournament:

- `void setupTournament()` - initializes player list and bracket structure
- `int simulateMatch(string player1, string player2)` - simulates match with random winner
- `void displayBracket()` - shows current tournament bracket using nested loops
- `void advanceRound()` - moves winners to next round with validation
- `string findChampion()` - determines tournament winner
- `void showStatistics()` - displays match statistics and player performance

Tournament Structure:

```
===== TOURNAMENT BRACKET SYSTEM =====
Enter number of players (must be power of 2): 8

Players: Alice, Bob, Carol, Dave, Eve, Frank, Grace, Henry

===== ROUND 1 (Quarter-Finals) =====
Match 1: Alice vs Bob → Winner: Alice
Match 2: Carol vs Dave → Winner: Dave
Match 3: Eve vs Frank → Winner: Frank
Match 4: Grace vs Henry → Winner: Grace

===== ROUND 2 (Semi-Finals) =====
Match 1: Alice vs Dave → Winner: Alice
Match 2: Frank vs Grace → Winner: Frank

===== ROUND 3 (Finals) =====
Championship: Alice vs Frank → Winner: Alice

🏆 TOURNAMENT CHAMPION: ALICE 🏆

Statistics:
Total Matches: 7
Alice: 3 wins, 0 losses
Frank: 2 wins, 1 loss
Dave: 1 win, 1 loss
Grace: 1 win, 1 loss
Bob: 0 wins, 1 loss
Carol: 0 wins, 1 loss
Eve: 0 wins, 1 loss
Henry: 0 wins, 1 loss
```

HARD QUESTION (6)

Question 6: Smart Home Automation Control System

Write a comprehensive C++ program that manages a smart home using functions and complex control structures:

System Functions Required:

1. `void initializeDevices()` - sets up all smart devices with default states
2. `void displayDashboard()` - shows current status of all devices using formatted output
3. `void controlLighting()` - manages smart lights with dimming and scheduling
4. `void manageClimate()` - handles temperature and humidity control with sensors
5. `void securitySystem()` - manages alarms, cameras, and access control
6. `void energyManagement()` - monitors and optimizes power consumption
7. `void createSchedule()` - sets up automated routines using time-based logic
8. `void processScenarios()` - executes predefined home scenarios
9. `void handleEmergencies()` - manages emergency protocols and notifications
10. `void generateReports()` - creates detailed usage and efficiency reports

Device Categories:

```cpp
// Smart Devices Array Structure
struct SmartDevice {
    string name;
    string category; // Lighting, Climate, Security, Entertainment
    bool isOn;
    int powerLevel;    // 0-100%
    string schedule;   // Time-based automation
    double energyUsage; // kWh consumed
};
```

Complex Control Logic Requirements:

- Use nested switch statements for device category and specific controls
- Implement time-based automation using loops and conditional logic
- Use multi-dimensional arrays for room-based device management
- Implement priority-based emergency protocols using if-else chains
- Use complex validation loops for user input and device compatibility

Sample Program Execution:

===== SMART HOME CONTROL SYSTEM =====
Current Time: 19:30, March 20, 2024
Outside Temperature: 18°C

===== HOME DASHBOARD =====
🏠 LIVING ROOM:
💡 Main Lights: ON (80%) | Schedule: Auto-dim at 21:00
🌡️ Thermostat: 22°C | Target: 21°C
📺 TV: OFF | Last used: 2 hours ago

🏠 BEDROOM:
💡 Bedside Lamps: OFF | Schedule: Auto-on at 22:00
🌡️ AC Unit: OFF | Energy Save Mode
🔒 Window Sensor: CLOSED

🏠 KITCHEN:
💡 Under-cabinet LEDs: ON (100%)
☕ Coffee Maker: SCHEDULED (06:30 tomorrow)
🚨 Smoke Detector: ACTIVE

===== CONTROL MENU =====
1. Lighting Control
2. Climate Management
3. Security System
4. Energy Management
5. Create Schedule
6. Activate Scenario
7. Emergency Protocols
8. System Reports
9. Exit

Enter choice: 6

===== HOME SCENARIOS =====
1. Good Morning (Lights on, Coffee start, News on TV)
2. Leaving Home (All off, Security armed, Temp lowered)
3. Movie Night (Dim lights, Sound system on, Climate adjust)
4. Bedtime (All lights off, Bedroom temp adjust, Lock doors)
5. Party Mode (All lights 100%, Music on, Climate boost)

Select scenario: 3

Activating Movie Night scenario...
✓ Dimming living room lights to 20%
✓ Turning off kitchen lights
✓ Setting thermostat to 20°C
✓ Activating surround sound system
✓ Closing automated blinds
✓ Disabling doorbell notifications

Movie Night scenario activated! Enjoy your film.

Enter choice: 8

```
===== SYSTEM REPORTS =====

Energy Usage (Last 24 hours):
Living Room: 12.5 kWh (45% of total)
Bedroom: 6.2 kWh (22% of total)
Kitchen: 5.8 kWh (21% of total)
Security System: 3.3 kWh (12% of total)
Total: 27.8 kWh

Device Efficiency Analysis:
Most Efficient: LED Lights (0.8 kWh/hour average)
Least Efficient: Old AC Unit (2.3 kWh/hour average)
Recommendation: Replace AC unit to save $180/year

Automation Performance:
Scheduled Events Today: 15
Successfully Executed: 14 (93%)
Failed: 1 (Coffee maker offline at 06:30)

Security Events:
Door Access: 4 entries (all authorized)
Motion Detected: 12 instances (living room)
Alerts Generated: 0
```

Advanced Implementation Features:

- Room-based device grouping with nested control structures
- Time-based conditional automation (if current time, then action)
- Energy optimization algorithms using comparison loops
- Emergency cascade protocols with priority-based execution
- Device compatibility checking using validation functions
- Usage pattern analysis with statistical calculations
- Predictive maintenance scheduling using device usage data
- Integration simulation with external services (weather, security)

Sample Complex Function:

```cpp
```

```
void processEmergency(string emergencyType) {
    if(emergencyType == "FIRE") {
        // Immediate actions using nested loops and conditions
        for(int room = 0; room < totalRooms; room++) {
            // Turn on all lights for evacuation
            for(int device = 0; device < devicesPerRoom; device++) {
                if(devices[room][device].category == "Lighting") {
                    devices[room][device].isOn = true;
                    devices[room][device].powerLevel = 100;
                }
            }
            // Unlock all doors, open windows if safe
            // Complex conditional logic for safety protocols
        }

        // Send alerts using loop for multiple contacts
        // Activate emergency broadcasting
        // Log all actions with timestamps
    }
    // Handle other emergency types with similar complexity
}
```