ECE 811 – SOFTWARE ENGINEERING

STRUCTURED PROGRAMMING – BEGINNERS'S STUDY GUIDE/REVISION

SECTION 1: CORE CONCEPTS

1. **What defines "structured programming"?**
   a) Using only machine code
   b) Exclusive focus on object-oriented design
   c) **Programs built using sequence, selection, iteration**
   d) Prioritizing low-level memory management
   Explanation: Structured programming emphasizes three control structures without unrestricted GOTO.

2. **The Böhm–Jacopini theorem proved that any program can be written using:**
   a) Objects and methods
   b) **Sequence, selection, iteration**
   c) Recursion only
   d) Parallel threads
   Explanation: This 1966 theorem established the theoretical foundation for structured programming.

3. **True/False: Modularity encourages dividing programs into independent, reusable functions.**
   Answer: True
   Explanation: Modularity is a core principle of structured programming.

4. **Which control structure is demonstrated here?**

```python
total = 0
for i in range(1, 6):
    total += i
```

a) Sequence
b) Selection
c) **Iteration**
d) Recursion
Explanation: The for loop repeats the addition operation.

5. **The "single entry/single exit" principle means:**
   a) Programs should have one input/output
   b) **Functions should have one entry and exit point**
   c) Variables must be initialized once
   d) Loops must run exactly once
   Explanation: This avoids complex control flows like multiple returns.

SECTION 2: CONTROL STRUCTURES

6. **Which is NOT a valid selection structure?**
   a) if-else
   b) switch-case
   c) goto label
   d) Ternary operator
   Explanation: goto violates structured programming principles.

7. **Convert this unstructured code to structured:**

```
10: IF X > 0 GOTO 30
20: PRINT "Negative"
25: GOTO 40
30: PRINT "Positive"
40: END
```

Answer:

```
if x > 0:
    print("Positive")
else:
    print("Negative")
```

Explanation: Removed GOTO using standard selection.

8. **What output results from this structured code?**

C-language

```
int a = 5;
while (a > 0) {
    printf("%d ", a);
    a--;
}
```

a) 5 4 3 2 1
b) 5 4 3 2 1 0
c) **5 4 3 2**
d) Infinite loop
Explanation: Loop runs for a=5,4,3,2,1 (stops when a=0).

9. **True/False: Deeply nested if-statements violate structured programming best practices.**
   Answer: True
   Explanation: Deep nesting reduces readability; refactor into functions.

SECTION 3: MODULARITY & DESIGN

10. **Ideal function length in structured programming is:**
    a) < 10 lines
    b) **< 30 lines**
    c) 50-100 lines
    d) No limit
    Explanation: Shorter functions enhance readability and maintainability.

11. **Top-down design involves:**
    a) Starting with hardware specs
    b) **Decomposing problems into sub-tasks**
    c) Optimizing for speed first
    d) Writing tests last
    Explanation: Break complex problems into simpler modules.

12. **Which function violates modularity?**

python

```python
# Option A
def calc_tax(income):
    rate = 0.2
    return income * rate
```

```python
# Option B
def process_data():
    global raw_data  # Uses global variable
    # ... 40 lines of code ...
```

Answer: Option B
Explanation: Relies on global state and is too long.

13. **Pseudocode for a modular design should emphasize:**
    a) **Task decomposition**
    b) Variable naming conventions
    c) Memory addresses
    d) Syntax details
    Explanation: Focuses on logical structure, not implementation.

SECTION 4: BENEFITS & PITFALLS

14. **Key benefit of structured programming:**
    a) Faster execution
    b) **Easier debugging**
    c) Lower memory usage
    d) Automatic parallelization
    Explanation: Linear flow simplifies error tracing.

15. **Common pitfall:**
    a) Too many small functions
    b) **Overusing global variables**
    c) Avoiding comments
    d) Using loops
    Explanation: Globals create hidden dependencies.

16. **True/False: Structured programming eliminates all bugs.**
    Answer: False
    Explanation: It reduces errors but doesn't eliminate them.

Section 5: Advanced Concepts

17. **Structured programming influenced:**
    a) Quantum computing
    b) **Object-oriented programming**
    c) Analog circuits
    d) Blockchain
    Explanation: OOP extends structured concepts with encapsulation.

18. **Dijkstra's paper "Go To Statement Considered Harmful" argued:**
    a) GOTO improves efficiency
    b) **GOTO creates unmaintainable code**
    c) All loops should use GOTO
    d) GOTO is essential for recursion
    Explanation: Seminal 1968 paper promoting structured alternatives.

19. **Recursion in structured programming:**
    a) Is prohibited
    b) **Must have a base case**
    c) Only works in functional languages
    d) Requires GOTO
    Explanation: Recursion is allowed but requires termination conditions.

SECTION 6: CODE ANALYSIS

Analyze this code:

C Language

```
float avg(int scores[], int n) {
    int sum = 0;                    // Line 1
    for (int i = 0; i < n; i++) {   // Line 2
        sum += scores[i];            // Line 3
    }                               // Line 4
    return (float)sum / n;           // Line 5
}
```

20. **Control structure in Line 2:**
    a) Sequence

b) Selection
c) **Iteration**
d) Recursion

21. **Violation of best practices?**
    a) No parameters
    b) **Missing input validation (n > 0?)**
    c) Using float
    d) Single exit point
    Explanation: Crashes if n=0 (division by zero).

22. **Refactor to improve modularity:**
    Answer: Split into sum_array() and avg().
    Explanation: Separate summation from averaging logic.

SECTION 7: HISTORICAL CONTEXT

23. **Earliest language designed for structured programming:**
    a) COBOL
    b) **ALGOL**
    c) Assembly
    d) LISP
    Explanation: ALGOL (1958) introduced block structures.

24. **Structured programming emerged as a response to:**
    a) Slow computers
    b) **"Spaghetti code" crisis**
    c) Internet security
    d) Cloud computing
    Explanation: 1960s software complexity required better paradigms.

SECTION 8: REAL-WORLD APPLICATION

25. **Why use structured programming in 2024?**
    a) **Maintainability of legacy systems**
    b) To impress interviewers
    c) It's obsolete
    d) Required for AI development
    Explanation: Core principles remain vital for maintainable code.

26. **Convert to structured code:**

```
READ X

IF X=0 GOTO END

PRINT X

GOTO Start

END: STOP
```

Answer:

python

```
while True:
    x = int(input())
    if x == 0:
        break
    print(x)
```

SECTION 9: BEST PRACTICES

27. **Which demonstrates good structured design?**
    a) 200-line function with nested loops
    b) validate_input() **+** process_data() **+** output_result()
    c) All variables global
    d) Functions named func1(), func2()
    Explanation: Modular separation of concerns.

28. **True/False: Comments should explain complex algorithms line-by-line.**
    Answer: False
    *Explanation: Code should be self-documenting; comments explain **why**.*

SECTION 10: COMPREHENSIVE

29. **Draw a flowchart for structured BMI calculator:**
    Answer:

    ```
    Start → Input height/weight → Validate → Calculate BMI → Classify → Output → End
    ```

    Explanation: Linear flow with single entry/exit points.

30. **Fix this unstructured code:**

C-Language

```
void printStatus(int score) {
    if (score > 50) goto PASS;
    printf("Fail");
    goto END;
    PASS: printf("Pass");
    END: return;
}
```

Answer:

C - Language

```
void printStatus(int score) {
    if (score > 50) {
        printf("Pass");
    } else {
        printf("Fail");
    }
}
```