# Advanced Debugging MCQs

## 1. Stack Overflow Due to Incorrect Recursion Base Case

Pseudo-Code:  
START   
 FUNCTION mystery(n)   
 IF n == 1 OR n == 2   
 RETURN 1   
 ELSE   
 RETURN mystery(n - 1) + mystery(n - 2)   
 END FUNCTION   
  
 PRINT mystery(-5)   
END

What will happen?

A) Prints a number   
B) Stack Overflow   
C) Syntax Error   
D) No Output

✅ Correct Answer: B) Stack Overflow

🔍 Explanation: The function lacks a proper base case for negative values. Calling `mystery(-5)` leads to an infinite recursive call, eventually causing a stack overflow.

## 2. Concurrency Issue in Multi-Threading

Pseudo-Code:  
START   
 x = 0   
  
 FUNCTION increment()   
 FOR i = 1 TO 1000   
 x = x + 1   
 END FOR   
 END FUNCTION   
  
 THREAD t1 = CALL increment()   
 THREAD t2 = CALL increment()   
 WAIT t1, t2   
  
 PRINT x   
END

What will be printed?

A) 2000   
B) Less than 2000   
C) More than 2000   
D) Syntax Error

✅ Correct Answer: B) Less than 2000

🔍 Explanation: Due to race conditions, both threads access and modify `x` concurrently without synchronization, leading to lost updates. This means the final value of `x` will be less than 2000.

## 3. Floating-Point Precision Edge Case

Pseudo-Code:  
START   
 x = 0.1 + 0.2   
 IF x == 0.3   
 PRINT "Equal"   
 ELSE   
 PRINT "Not Equal"   
END

What will be printed?

A) "Equal"   
B) "Not Equal"   
C) Syntax Error   
D) Runtime Error

✅ Correct Answer: B) "Not Equal"

🔍 Explanation: Due to floating-point precision errors, `0.1 + 0.2` does not exactly equal `0.3`. Instead, it results in 0.30000000000000004, causing the condition to fail.

## 4. Memory Leak Due to Improper Object Management

Pseudo-Code:  
START   
 CLASS Node   
 VAR data   
 VAR next   
  
 FUNCTION createList()   
 HEAD = NEW Node()   
 TEMP = HEAD   
  
 FOR i = 1 TO 5   
 TEMP.next = NEW Node()   
 TEMP = TEMP.next   
 END FOR   
 END FUNCTION   
  
 CALL createList()   
END

What will happen?

A) Program executes normally   
B) Memory Leak   
C) Syntax Error   
D) Runtime Error

✅ Correct Answer: B) Memory Leak

🔍 Explanation: The function `createList()` dynamically allocates memory for linked list nodes but never deallocates them before returning. Since `HEAD` is a local variable and is not returned, all allocated nodes become unreachable, leading to a memory leak.