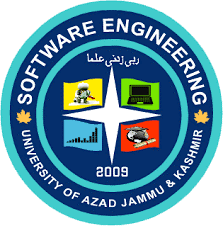
****

**The University of Azad Jammu and Kashmir**

**Lab Task # 09**

**Course Instructor:** Engr. Saba Rafique **Semester:** Fall-2024

**Session:** 2022-2026 **Submission Date:** Jan 13, 2024

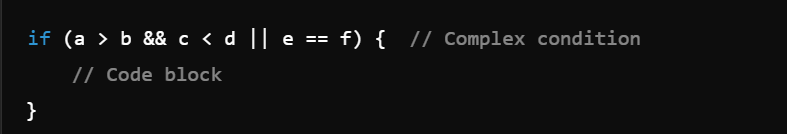
**Submitted By:** Syeda Urwa Ajmal **Roll No:** 2022-SE-16

**Course Name:** SC&D **Code:** SE-3102

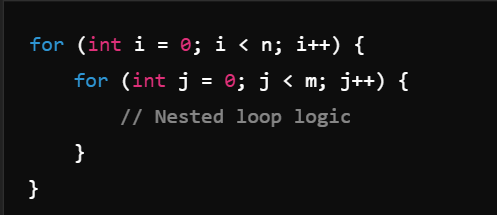
# Conceptual Questions

**Conceptual Questions:**

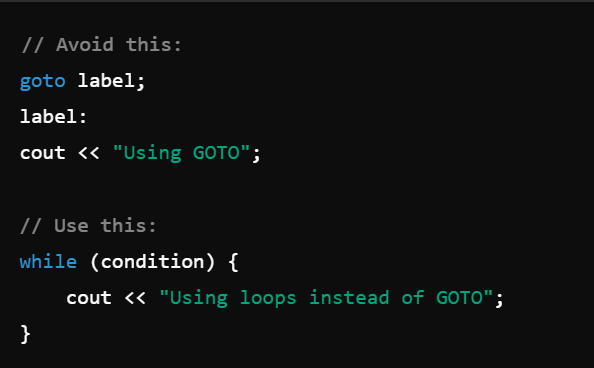
1. **When are conditional statements hard to read?**
   * Conditional statements become hard to read when:
     + There are too many nested if-else blocks.
     + The conditions are overly complex and not modularized.
     + Poorly named variables make the logic unclear.
   * **Example:**



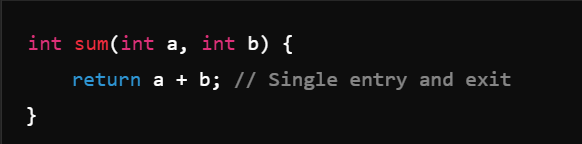
1. **When are repetitive loops hard to read?**
   * Repetitive loops become hard to read when:
     + They are deeply nested.
     + Lack of clear iteration logic or poorly named variables.
     + Unnecessary operations are performed inside the loop.
   * **Example:**



1. **Why is the use of the GOTO discouraged? What constructs replace the GOTO?**
   * **Reason:**
     + GOTO creates "spaghetti code," making it difficult to follow logic and debug.
   * **Replacement:**
     + Use for, while loops, and function calls instead.
   * **Example:**



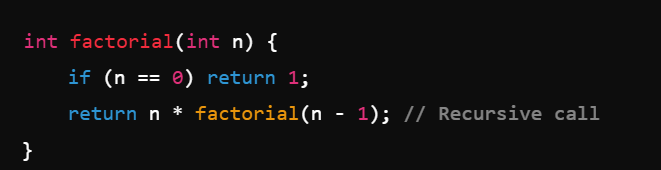
1. **What advantage(s) does design with only single entry and single exit points provide?**
   * Advantages:
     + Easier to debug and test.
     + Improves code readability and maintainability.
   * **Example:**



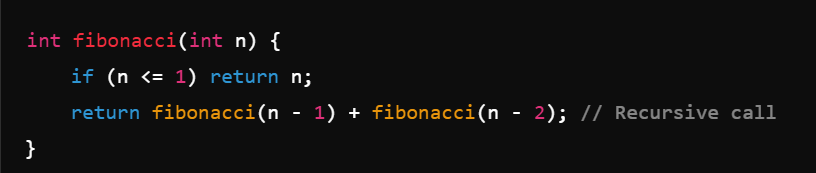
1. **How can the validity of a simplified Boolean expression be verified?**
   * **Verification Method:**
     + Use truth tables or logical equivalence rules.
   * **Example:**



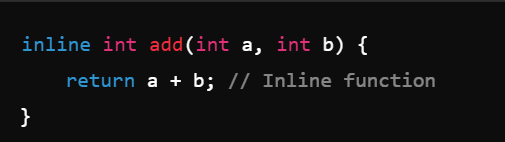
1. **What drawbacks are associated with recursion?**
   * **Drawbacks:**
     + High memory usage due to stack frames.
     + Risk of stack overflow for deep recursion.
   * **Example:**



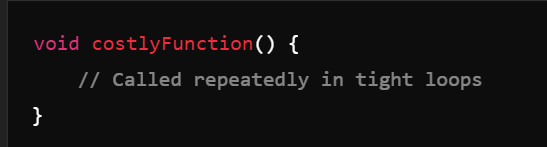
1. **When would recursion be preferred to iteration?**
   * **Use Cases:**
     + When the problem is naturally recursive (e.g., tree traversal).
   * **Example:**



1. **What does function inlining do? What is its effect?**
   * **Function Inlining:**
     + Replaces a function call with the actual code of the function.
   * **Effect:**
     + Reduces function call overhead but increases binary size.
   * **Example:**



1. **When is a function call too costly? Why?**
   * **Reasons:**
     + When it involves deep recursion, causing excessive memory usage.
     + Frequent calls with high overhead in performance-critical sections.
   * **Example:**



1. **List some common best practices for design and data declarations.**
   * **Best Practices:**
     + Use meaningful variable names.
     + Keep functions small and focused.
     + Avoid global variables.
     + Use comments for clarity.
     + Follow consistent coding standards.