**BATTLE OF BUGS**

**Event Plan: Battle of Bugs Debugging Event**

**Event Date**: 27th September 2024  
**Venue**: Lab 6, 4th Floor, BMS  
**Start Time**: 9:00 AM

**Event Structure:**

The 'Battle of Bugs' is a competitive debugging event, designed for a total of 16 teams, with each team consisting of 2 members. The event will have multiple rounds, where teams will face off in a 2 vs 2 format. Each round will determine which teams advance further, with a focus on real-time debugging challenges.

**Event Timeline:**

* **9:00 am**: Participant Check-in and Setup
* **9:30 am**: **First Round Begins**
  + 8 teams vs 8 teams. Each team competes against another team in a debugging challenge.
  + 8 winning teams will move to the second round. The remaining 8 teams will be eliminated.
* **10:30 am**: **Second Round Begins**
  + 4 teams vs 4 teams. The winners from the first round will again face off against each other.
  + 4 teams will advance to the final round. The other 4 teams will be eliminated.
* **11:30 am**: **Final Round Begins**
  + 2 vs 2 teams. The final four teams will compete for the top positions.
  + 2 winning teams will secure the top places (1st and 2nd), while the other 2 will be eliminated.

**Conclusion:**

By the end of the final round, the top 2 teams will be determined based on their performance in debugging and problem-solving. Certificates and prizes will be awarded to the winners, recognizing their achievement.

### 

### Event Questions Format Plan

**Round 1: Basic Debugging (9:30 am)**

* Focus: Syntax errors, logical errors, runtime errors.
* 8 teams compete (2 vs 2 format).
* Time Limit: 30 minutes.
* Outcome: 8 teams advance, 8 teams disqualified.

**Round 2: Intermediate Debugging (10:30 am)**

* Focus: Algorithmic mistakes, infinite loops, memory leaks, multi-threading issues.
* 4 teams advance (2 vs 2 format).
* Time Limit: 45 minutes.
* Outcome: 4 teams advance, 4 teams disqualified.

**Final Round: Participant-Designed Challenges (11:30 am)**

* **Concurrency Issues**: Fix race conditions or synchronization problems.  
  Example: Multiple threads accessing a shared bank account causing inconsistent balances.
* **Performance Bottlenecks**: Optimize inefficient algorithms with high time complexity.  
  Example: A function running in O(n²) instead of O(n log n).
* **Complex Data Structures**: Debug issues with data structures like unbalanced trees.  
  Example: Fix the insertion logic in a binary tree to maintain balance.
* **Obfuscated Code**: Fix multiple bugs in a large, complex codebase.  
  Example: A file-handling program that crashes due to memory and logic errors.

**Participation Certificates**: Awarded to all 16 teams.