Republic of the Philippines

University of Cabuyao

(Pamantasan ng Cabuyao)

COLLEGE OF COMPUTING STUDIES

Katapatan Mutual Homes, Brgy. Banay-banay, City of Cabuyao, Laguna 4025

FINAL LABORATORY EXAMINATION

ITP109 – PLATFORM TECHNOLOGIES

2nd Semester S.Y. 2024-2025

NAME: ____________________ SCORE: __________

STUDENT NUMBER: ____________________ SECTION: __________

NAME OF FACULTY: ____________________ DATE: __________

Banker's Algorithm System – A Scenario Simulation

TechOS Inc. Operating System Resource Management

At TechOS Inc., a leading innovator in operating systems, a new project was underway—one that demands absolute precision in resource management. You, a seasoned Systems Engineer, was entrusted with a mission critical to the stability of the company's latest OS: implementing the Banker's Algorithm. This algorithm would serve as a gatekeeper, ensuring the system never drifted into a deadlocked state by dynamically assessing and managing resource allocation in real time.

To accomplish this, two simulation models must be developed, each offering a unique perspective on how the system responds to varying process demands:

1. **Named Resource-Based Simulation (Object-Oriented Approach)**
2. **Matrix-Based Simulation (Classical Vector/Matrix Approach)**

Each simulation was to meet a strict set of criteria: it needed to handle resource allocation and request management, verify system safety, avoid deadlock, and—if needed—initiate recovery processes while logging every decision.

Rubric for Banker's Algorithm System – Scenario Simulation

Criteria	Excellent (10 pts)	Good (8 pts)	Fair (6 pts)	Needs Improvement (2 pts)	Points
1. Implementation of Banker's Algorithm	Correct and complete implementation for both models; handles all scenarios accurately.	Minor issues, but core logic works and avoids deadlocks.	Incomplete or inconsistent logic; handles limited cases.	Fails to prevent deadlocks; core functionality missing or broken.	/10
2. Two Simulation Models	Both Object-Oriented and Matrix-Based models are clearly implemented and well-differentiated.	Both models implemented but some structural overlap or confusion.	One model is incomplete or not functional; unclear separation of approaches.	Only one model exists or both are poorly implemented.	/10
3. Resource Allocation & Request Handling	Fully supports dynamic requests, allocations, and user input with robust error checking.	Functional in most cases; some validation or logic issues present.	Limited flexibility or checks; allocation logic occasionally fails.	Handles few cases or frequently errors during request/allocation.	/10
4. System Safety & Deadlock Avoidance	Consistently checks for system safety; deadlocks are accurately avoided or reported.	Safety checks are mostly correct; rare deadlock scenarios may occur.	Deadlock detection unreliable or partial; safety logic is weak.	No effective safety or deadlock handling mechanisms.	/10

5. Logging & Decision Explanation	Clearly logs all key actions with meaningful comments or outputs.	Logging is present and mostly helpful; some decisions unclear.	Basic logs with little explanation; difficult to follow program flow.	No or confusing logs; decisions are not explained.	/10
6. Code Structure & Readability	Code is modular, clean, and well-documented with clear naming and structure.	Mostly clean with good use of functions or classes; some repetition.	Code is readable but messy or under-commented.	Disorganized code; unreadable or poorly documented.	/10
7. Recovery Mechanism (if used)	Recovery process is implemented, clear, and re-stabilizes the system effectively.	Present but simplistic; may not handle all unsafe states.	Incomplete recovery approach or rarely invoked.	No recovery or faulty mechanism.	/10
8. Input Flexibility & Testing Support	Accepts varied inputs (manual/file-based); well-tested with edge cases.	Accepts most inputs; some test cases fail or unsupported formats.	Limited input handling or few test scenarios covered.	Input must be hardcoded; testing is minimal or absent.	/10
9. Innovation & Enhancement Features	Goes beyond requirements (GUI, animations, advanced interactivity or modularity).	Includes some enhancements beyond basic logic.	Slight enhancements, but mostly standard implementation.	Basic logic only; no enhancements present.	/10
10. Overall Execution & Reliability	Highly reliable system with smooth performance and meaningful feedback.	Generally reliable; minor bugs or inconsistencies.	Occasionally fails or gives unclear results.	Frequent failures; unreliable system behavior.	/10

Total Score: **/100**

Prepared by:	Date:	Checked and Reviewed by:	Date:	*Approved by:*	Date:
MR. RENZO EVANGELISTA Professor – CCS **Mr. Philip Redondo** MR. PHILIP REDONDO Professor - CCS	May 6, 2025	Prof. Arcelito Quitachon Program Chair, BSIT	May 6, 2025	DR. Gina Montecillo Dean	