# Department of Computer Technology

### Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

### Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem- solving skills through emerging technologies**.**

## Session 2025-2026

**Mission:** Means to achieve Vision

**Vision:** Dream of where you want.

**Program Educational Objectives of the program (PEO):** (broad statements that describe the professional and career accomplishments)

|  |  |  |  |
| --- | --- | --- | --- |
| PEO1 | **Preparation** | **P: Preparation** | **Pep-CL abbreviation**  **pronounce as Pep-si-lL easy to recall** |
| PEO2 | **Core Competence** | **E: Environment (Learning Environment)** |
| PEO3 | **Breadth** | **P: Professionalism** |
| PEO4 | **Professionalism** | **C: Core Competence** |
| PEO5 | **Learning**  **Environment** | **L: Breadth (Learning in diverse areas)** |

**Program Outcomes (PO):** (statements that describe what a student should be able to do and know by the end of a program)

## Keywords of POs:

Engineering knowledge, Problem analysis, Design/development of solutions, Conduct Investigations of Complex Problems, Engineering Tool Usage, The Engineer and The World, Ethics, Individual and Collaborative Team work, Communication, Project Management and Finance, Life-Long Learning

**PSO Keywords:** Cutting edge technologies, Research

“I am an engineer, and I know how to apply engineering knowledge to investigate, analyse and design solutions to complex problems using tools for entire world following all ethics in a collaborative way with proper management skills throughout my life.” *to contribute to the development of cutting-edge technologies and Research*.

**Integrity:** I will adhere to the Laboratory Code of Conduct and ethics in its entirety.

## Name and Signature of Student and Date

(Signature and Date in Handwritten)

# Department of Computer Technology

### Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

### Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem- solving skills through emerging technologies**.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Session** | **2025-26 (ODD)** | **Course Name** | **Operating System Lab** |
| **Semester** | **5** | **Course Code** | **23IOT1504** |
| **Roll No** | 71 | **Name of Student** | Vedant Yerne |

|  |  |
| --- | --- |
| Practical Number |  |
| Course Outcome | 1. Understand Computer System Configuration and Simulate system resources efficiently using Linux Commands (CO1) 2. Analyse operating system functionalities utilizing system calls, thread programming and process scheduling   algorithms (CO2)   1. Apply Synchronization primitives to implement a Deadlock-free solution(CO3) 2. Simulate Disk scheduling, Memory allocation, File allocation, page replacement algorithms (CO4) |
| Aim | Stimulate banks deadlock algorithm |
| Problem Definition | Design and implement a simulation of the **Banker's Algorithm** for deadlock avoidance. The system should safely allocate resources to multiple processes without entering a deadlock state. The program must check whether a requested resource allocation can be granted without compromising system safety and display the safe sequence if it exists. |
| Theory  (100 words) | Banker’s Algorithm, proposed by Edsger Dijkstra, is used in operating systems to avoid deadlocks by simulating resource allocation for processes in a way that ensures a safe state. Each process declares the maximum number of instances of each resource it may need. The system only allocates requested resources if doing so keeps it in a safe state — one where all processes can complete without causing deadlock. The algorithm uses key matrices: Allocation, Maximum, Need (Maximum - Allocation), and Available. A request is granted only if it doesn't exceed the declared need and enough resources are available, ensuring safe execution. |
| Procedure and Execution  (100 Words) | Step for Implementation:   **Input the number of processes and resource types.**   **Input the Allocation matrix** (resources currently allocated to each process).   **Input the Maximum matrix** (maximum resources each process may need).   **Calculate the Need matrix = Maximum - Allocation.**   **Input the Available resources.**   **Run the Banker's safety algorithm:**   * Find a process whose needs can be met with available resources. * Pretend to allocate those resources and mark the process as finished. * Release its resources back to available. * Repeat until all processes are finished or no further progress is possible.    **If all processes finish**, the system is in a safe state and the safe sequence is printed.   **If not**, the system is in an unsafe state (potential deadlock). |
| Code:  #include <stdio.h>  #include <stdbool.h>  int main() {  int n, m;  printf("Enter number of processes: ");  scanf("%d", &n);  printf("Enter number of resource types: ");  scanf("%d", &m);  int alloc[n][m], max[n][m], need[n][m], avail[m];  // Input Allocation Matrix  printf("\nEnter Allocation Matrix:\n");  for (int i = 0; i < n; i++)  for (int j = 0; j < m; j++)  scanf("%d", &alloc[i][j]);  // Input Maximum Matrix  printf("\nEnter Maximum Matrix:\n");  for (int i = 0; i < n; i++)  for (int j = 0; j < m; j++)  scanf("%d", &max[i][j]);  // Input Available Resources  printf("\nEnter Available Resources:\n");  for (int j = 0; j < m; j++)  scanf("%d", &avail[j]);  // Calculate Need Matrix = Max - Allocation  for (int i = 0; i < n; i++)  for (int j = 0; j < m; j++)  need[i][j] = max[i][j] - alloc[i][j];  bool finish[n];  for (int i = 0; i < n; i++)  finish[i] = false;  int safeSequence[n];  int work[m];  for (int i = 0; i < m; i++)  work[i] = avail[i];  int count = 0;  while (count < n) {  bool found = false;  for (int i = 0; i < n; i++) {  if (!finish[i]) {  bool canRun = true;  for (int j = 0; j < m; j++) {  if (need[i][j] > work[j]) {  canRun = false;  break;  }  }  if (canRun) {  for (int k = 0; k < m; k++)  work[k] += alloc[i][k];  safeSequence[count++] = i;  finish[i] = true;  found = true;  }  }  }  if (!found) {  printf("\nSystem is in UNSAFE state (deadlock may occur).\n");  return 1;  }  }  // If system is in a safe state  printf("\nSystem is in SAFE state.\nSafe sequence is: ");  for (int i = 0; i < n; i++)  printf("P%d ", safeSequence[i]);  printf("\n");  return 0;  } |
| Output: |

# Department of Computer Technology

### Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

### Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem- solving skills through emerging technologies**.**

|  |  |
| --- | --- |
|  |  |
| Output Analysis |  The system receives input for **5 processes (P0 to P4)** and **3 resource types**.   Each process provides its **Allocation** (resources currently held) and **Maximum** (maximum resources it may need).   The program calculates the **Need matrix** by subtracting Allocation from Maximum for each resource per process.   The system starts with the given **Available resources**: A = 3, B = 3, C = 2.   Using the Banker's Algorithm, it checks if any process's **Need** can be satisfied with the current **Available** resources.   It successfully finds a **safe sequence** in which all processes can execute without causing a deadlock.   The safe sequence found is: **P1 → P3 → P4 → P0 → P2**.   This means the system is in a **SAFE state**, and all processes can complete by sequentially allocating and releasing resources.   No deadlock will occur if processes request resources as declared in their **Maximum matrix**. |
| Link of student Github profile where lab assignment has been uploaded |  |
| Conclusion | The implementation and simulation of the **Banker's Algorithm** successfully demonstrate how a system can avoid deadlocks by carefully analyzing resource allocation and ensuring it remains in a **safe state**. By calculating the **Need**, **Available**, and using a safety check, the algorithm ensures that all processes can complete without waiting indefinitely for resources. The safe sequence output confirms that resource allocation can proceed without causing a deadlock. This algorithm is crucial in multi-processing environments where resource management and system stability are priorities. Therefore, **Banker's Algorithm serves as an effective strategy for deadlock avoidance** in operating systems. |
| Plag Report (Similarity index < 12%) |  |
| Date | 22/0902025 |