**The University of Azad Jammu and Kashmir, Muzaffarabad**



|  |  |
| --- | --- |
| **Submitted to:** | **Engr. Sidra Rafique** |
| **Course Title:** | **Operating System** |
| **Course Code:** | **SE-3205** |
| **Session:** | **2022-26** |
| **Semester:** | **6th** |
| **Roll No:** | **2022-SE-01** |
| **Submitted from:** | **Khurram Farman** |
| **Lab Task No** | **12** |
|  |  |

**Bachelors of Science in Software Engineering (2022-26)**

**Department of Software Engineering**

# Banker’s Algorithm Report

## Introduction

This report presents the implementation and analysis of the **Banker’s Algorithm** for deadlock avoidance in operating systems. The algorithm determines whether the system is in a **safe or unsafe state** by carefully allocating resources only when they do not risk causing a deadlock.

## Objectives

* To understand the working of the Banker’s Algorithm.
* To calculate **Need Matrix** and check for safe sequence generation.
* To evaluate system states (Safe vs Unsafe) based on resource allocation.

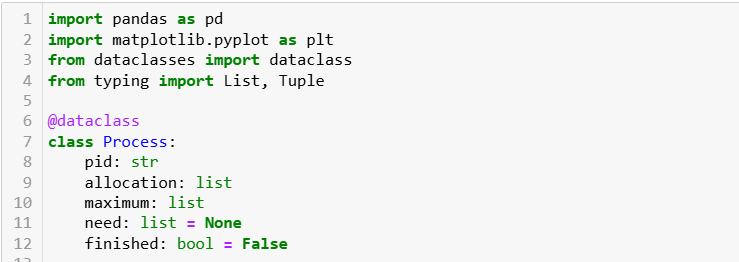
## Methodology

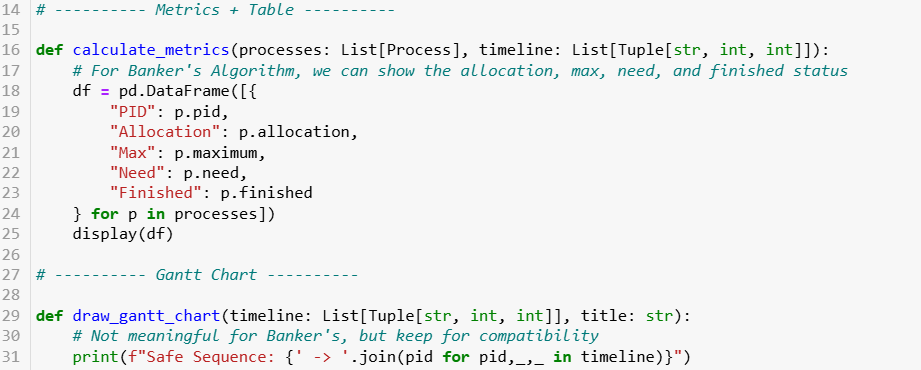
The algorithm was implemented in **Python using Jupyter Notebook**.

Steps followed:

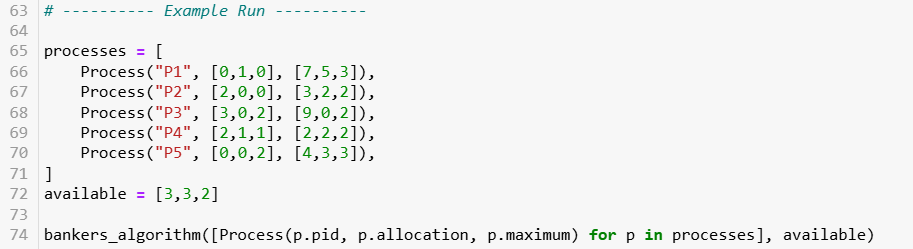
1. Define processes with their **Allocation** and **Maximum** resource requirements.
2. Compute the **Need Matrix** = Maximum – Allocation.
3. Initialize the **Available Vector** to represent current free resources.
4. Iteratively check each process:
   * If its needs ≤ available resources, mark it **finished**, release its allocated resources back to the pool, and append to the safe sequence.
5. Continue until either:
   * All processes are finished (Safe State).
   * Or no further allocation is possible (Unsafe State).
6. Display results in a **pandas DataFrame** and print the safe sequence.

## Code Implementation

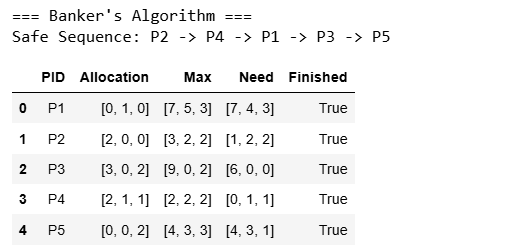








## Output



## Results and Discussion

* The algorithm successfully detected a **Safe State**.
* A valid **Safe Sequence** (P2 → P4 → P1 → P3 → P5) was generated.
* The **Need Matrix** helps determine if each process can finish with current resources.
* If no process satisfies its needs, the system is in an **Unsafe State** (deadlock possible).

## Conclusion

* The **Banker’s Algorithm** ensures system safety by allocating resources only when a safe sequence exists.
* It prevents deadlocks but may deny requests even when deadlock is not immediately possible, leading to **resource underutilization**.
* In this case, the system was safe and all processes were able to finish in sequence.
* Overall, Banker’s Algorithm provides a **practical balance between safety and resource utilization** in multiprogramming systems.