Banker's Algorithm Code Features Report

# Introduction

This document provides a detailed explanation of the features implemented in the C++ program for the Banker's Algorithm, which is used to manage resource allocation and avoid deadlock in a multi-process system. The code provides several options for users to interact with the system, including initialization methods, safety checks, and resource request simulations.

# Features Overview

## 1. Initialization Mode Selection

When the program starts, the user is prompted to choose one of the following initialization modes:  
1. Use default values  
2. Use random values  
3. Enter values manually

Each option serves a distinct purpose:  
1. The default values are pre-set matrices that are commonly used in examples.  
2. Randomly generates `Maximum`, `Allocation`, and `Available` matrices for a unique simulation.  
3. Prompts the user to manually input all matrices for a custom configuration.

## 2. Matrix Initialization

The program uses three key matrices: `Allocation`, `Maximum`, and `Available`.  
These matrices represent the current allocated resources, the maximum demand of each process, and the available resources in the system. The user can choose between initializing them manually, with default values, or with random values.

## 3. Need Matrix Calculation

The `Need` matrix is calculated as:  
Need[i][j] = Maximum[i][j] - Allocation[i][j]  
This matrix represents the remaining resources that each process might still need in order to complete its task.

## 4. Safety Algorithm

The program checks if the system is in a safe state using the safety algorithm. It tries to find a sequence of processes that can finish without causing a deadlock. If all processes can finish, the system is considered safe; otherwise, it is unsafe.

## 5. Resource Request Option

The user can simulate a resource request by entering a process ID and the number of resources requested. The program then checks if the request is valid (i.e., does not exceed the need or available resources). If the request can be granted and the system remains safe, it is approved; otherwise, it is denied.

## 6. Request Validation and Rollback

If a resource request exceeds the need or available resources, the system will output an error message. If granting the request would result in an unsafe state, the allocation is rolled back to maintain system safety.

# Summary of Features

|  |  |
| --- | --- |
| Feature | Description |
| Initialization Mode |  |
|  | Choose default, random, or manual matrices |
| Safety Check |  |
|  | Determines if system is in a safe state |
| Resource Request |  |
|  | Simulate dynamic resource requests |
| Rollback |  |
|  | Reverts changes if unsafe state would result |
| Matrices Printing |  |
|  | Displays Allocation, Maximum, and Need matrices |
| Random Mode |  |
|  | Generates random cases for simulation |