DEADLOCK AVOIDANCE

CAPSTONE PROJECT REPORT

CSA0405 - OPERATRING SYSTEMS OF FILE SYSTEMS IMPLEMENATATION

Under the guidance of

DR. Yuvarani

SIMATS ENGINEERING

THANDALAM

MARCH 2024

Submitted by

K.L.V JAYARAM [Reg No:192210314]
M. MAHESH [Reg No:192210312]
T. ANOOP CHANDAR [Reg No:192210244]

BONAFIDE CERTIFICATE

Certified that this project report titled "**DESIGNING AN AUDIOBOOK USING PYTHON"** is the Bonafide work of **"K.L.V JAYARAM [192210314], M. MAHESH [192210312], T. ANOOP CHANDAR [Reg No:192210244]"** who carried out the project work under my supervision as a batch. Certified further, that to the best of my knowledge the work reported herein does not form any other project report.

Date: Project Supervisor Head of the Department

PROBLEM STATEMENT:

The problem statement focuses The Banker's Algorithm addresses resource allocation and deadlock avoidance in operating systems. It manages finite resources across various types, accommodating multiple processes vying for these resources. By assessing current and future resource demands of processes, it selectively grants requests to maintain system safety, ensuring that all processes can complete without encountering deadlock. Its objective is to optimize resource utilization while mitigating the risk of deadlock occurrence. It also helps the OS to successfully share the resources between all the processes. It is called the banker's algorithm because bankers need a similar algorithm- they admit loans that collectively exceed the bank's funds and then release each borrower's loan in installments. The banker's algorithm uses the notation of a safe allocation state to ensure that granting a resource request cannot lead to a deadlock either immediately or in the future. In other words, the bank would never allocate its money in such a way that it can no longer satisfy the needs of all its customers. The bank would try to be in a safe state always.

PROPOSED DESIGN WORK:

1.KEY COMPONENTS

- Available Resources
- Maximum Demand Matrix
- Allocation Matrix
- Need Matrix
- Safety Algorithm
- Request and Release Mechanisms
- Resource Utilization
- Performance Evaluation and Testing
- User Interaction and Error Handling
- Visualization and Reporting
- Adaptability and Scalability
- Optimal Task Sequence Planning
- Real-time Scheduling Considerations

2.FUNCTIONALITY

Resource Allocation: It facilitates the allocation of resources to processes in a way that prevents deadlock. The algorithm ensures that resources are only allocated if the system will remain in a safe state, meaning that all processes can eventually complete their execution without getting stuck in a deadlock situation.

Deadlock Avoidance: By carefully tracking the current allocation, maximum demand, and available resources, the Banker's Algorithm effectively avoids deadlock by never allowing a state to occur where all processes are simultaneously holding resources and

waiting for additional resources held by other processes. It evaluates resource requests dynamically, considering the overall system state to prevent the possibility of deadlock.

Safe State Detection: The primary functionality of the Banker's Algorithm is to detect and maintain a "safe state" in the system. A safe state is a state where the system can allocate resources to processes in such a way that no deadlock will occur. The algorithm employs various techniques to determine whether a state is safe or not.

Resource Request Handling: When a process requests additional resources, the Banker's Algorithm checks whether granting those resources will lead to a safe state. It evaluates whether the process's request can be granted immediately based on the available resources and the potential future resource requests of other processes.

Dynamic Resource Allocation: The algorithm dynamically adjusts resource allocation based on the current state of the system. It constantly updates its assessment of the available resources and the resource needs of processes to ensure that resources are allocated efficiently while preventing deadlock.

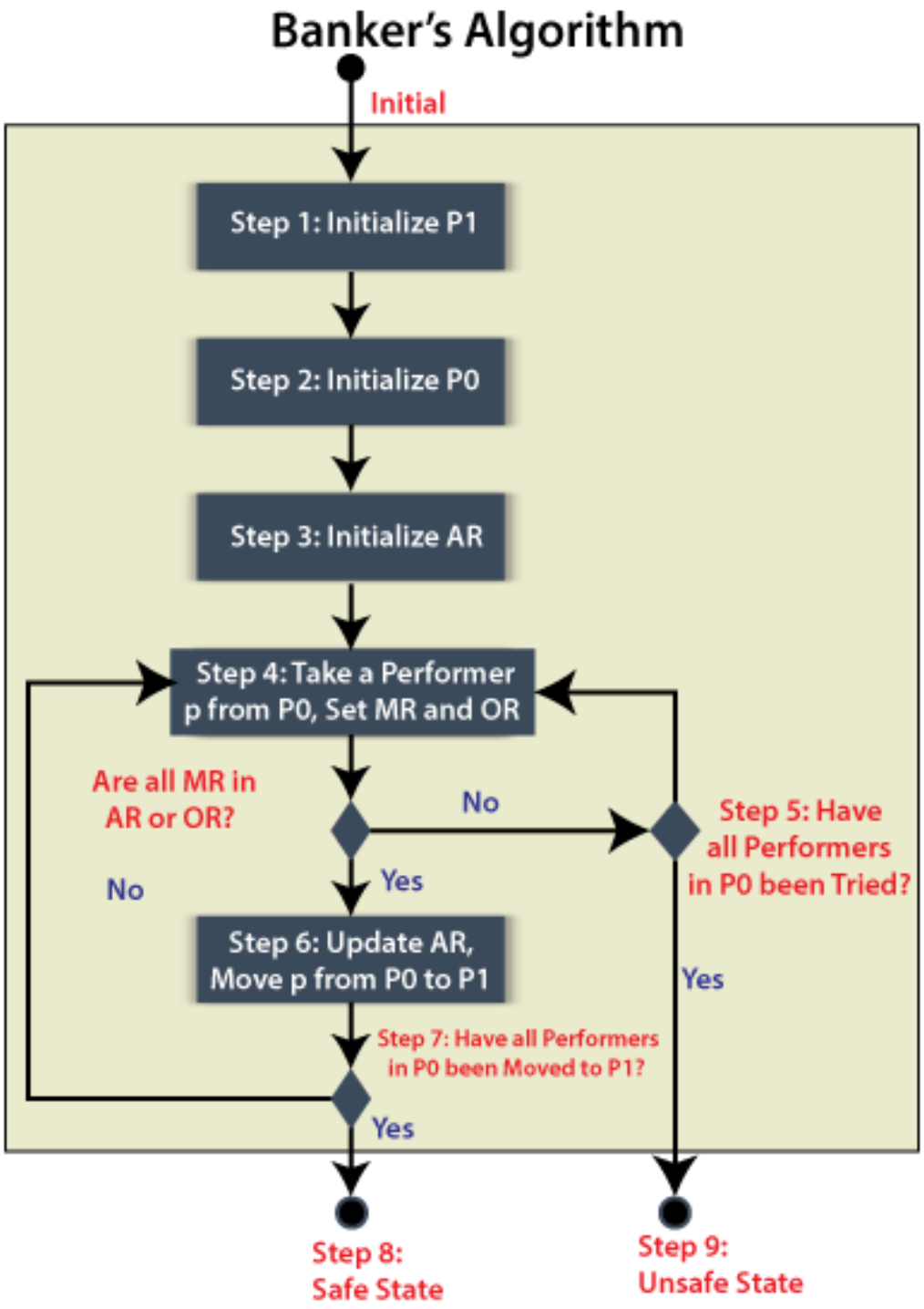
Resource Release Handling: When a process releases resources, the Banker's Algorithm updates its internal data structures to reflect the change in resource availability. It may then reevaluate the resource allocation to determine if additional processes can be granted resources without compromising system safety.

Optimization of Resource Utilization: While ensuring deadlock avoidance, the Banker's Algorithm also aims to optimize resource utilization. It tries to allocate resources in a way that maximizes the number of processes that can execute concurrently without violating safety constraints.

Process Suspension and Resumption: In cases where granting a resource request would lead to an unsafe state, the Banker's Algorithm may temporarily suspend the requesting process until sufficient resources become available. Once the system enters a safe state again, the suspended process can be resumed and granted the necessary resources.

Overall, the Banker's Algorithm provides a systematic approach to resource allocation and deadlock avoidance in operating systems, ensuring both system safety and efficient resource utilization.

3.ARCHITECTURAL DESIGN



UI DESIGN

1.LAYOUT DESIGN

- Scheduling Module
  - Algorithm Implementation
  - Optimization Submodule
- Resource Management
  - Resource Allocation
  - Resource Monitoring Subsystem
- Process Management
  - Process Handling Submodule
  - Task Queue Management
- User Interface and Reporting
  - User Interaction Module
  - Visualization and Reporting
- Performance Evaluation
  - Testing and Validation Subsystem
- Adaptability and Scalability
  - System Adaptation Submodule
  - Scalability Management
- Error Handling and Recovery
  - Error Detection and Handling
  - Recovery Subsystem

2.FEASIBLE ELEMENTS USED

- **ELEMENTS POSITIONING :**

- Introduction to Deadlock Avoidance
  - Brief overview of the importance and benefits of Deadlock Avoidance in operating systems.
- Bankers Algorithms Implementation
  - Detailed explanation of Bankers Algorithm.
- Optimization Techniques
  - Strategies and methods used to enhance algorithm performance.
- Resource Management
  - Allocation and utilization of resources to optimize processing efficiency.
- Performance Evaluation and Testing
  - Importance of testing and evaluating the effectiveness of bankers algorithms.

- User Interaction and Error Handling
    - Ensuring user-friendly interfaces and effective error handling mechanisms.
- Visualization and Reporting
    - Providing visual representations and detailed reports for analysis and decision-making.
- Adaptability and Scalability
    - Design considerations for adapting to varying workloads and scaling the system.

3.ELEMENTS FUNCTION

1. Deadlock Avoidance Algorithm:
    - Bankers Algorithm
2. Resource Utilization:
    - Efficient CPU resource allocation
    - Minimization of resource wastage
3. Process Management:
    - Effective handling of process arrival times
    - Burst times, and priority values
4. Performance Evaluation:
    - Rigorous testing to validate algorithm efficiency
    - Assessment of system responsiveness
5. User Interaction and Error Handling:
    - User-friendly interfaces
    - Error handling mechanisms
    - Feedback for users
6. Visualization and Reporting:
    - Visualizations of scheduling results
    - Comprehensive reports for analysis
7. Adaptability and Scalability:
    - Adaptable to different workloads
    - Scalable to accommodate increasing processes
8. Real-time Scheduling Considerations:
    - Incorporation of real-time constraints to meet critical task deadlines.

CODE AND OUTPUT:

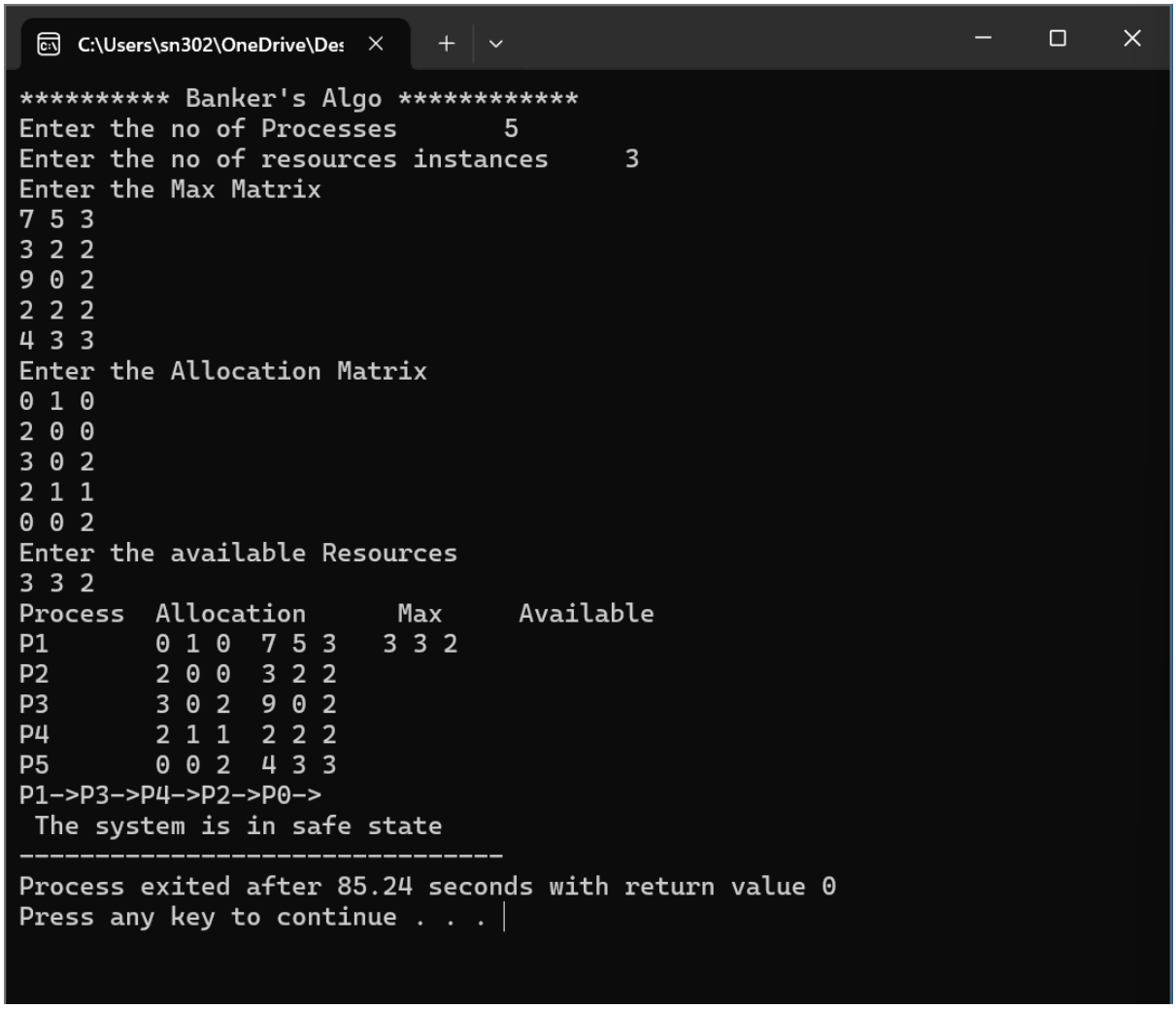
CODE

```
#include<stdio.h>
#include<conio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n,r;

void input();
void show();
void cal();
int main(){
int i,j;
printf("********** Banker's Algo ************\n");
input();
show();
cal();
getch();
return 0;
}

void input(){
        int i,j;
        printf("Enter the no of Processes\t");
        scanf("%d",&n);
        printf("Enter the no of resources instances\t");
        scanf("%d",&r);
        printf("Enter the Max Matrix\n");
        for(i=0;i<n;i++){
                for(j=0;j<r;j++){
                        scanf("%d",&max[i][j]);
                                }
                }
        printf("Enter the Allocation Matrix\n");
        for(i=0;i<n;i++){
                for(j=0;j<r;j++){
                        scanf("%d",&alloc[i][j]);
                }
        }
        printf("Enter the available Resources\n");
        for(j=0;j<r;j++){
        scanf("%d",&avail[j]);
        }}
```

```
void show(){
	int i,j;
	printf("Process\t Allocation\t Max\t Available\t");
	for(i=0;i<n;i++){
	printf("\nP%d\t ",i+1);
	for(j=0;j<r;j++){
	printf("%d ",alloc[i][j]);
	}
	printf("\t");
	for(j=0;j<r;j++){
	printf("%d ",max[i][j]);
	}
	printf("\t");
	if(i==0)
	{
	for(j=0;j<r;j++)
	printf("%d ",avail[j]);
	}}
}

void cal(){
	int finish[100],temp,need[100][100],flag=1,k,c1=0;
	int safe[100];
	int i,j;
	for(i=0;i<n;i++){
	finish[i]=0;
	}
	for(i=0;i<n;i++){
	for(j=0;j<r;j++)
	{
	need[i][j]=max[i][j]-alloc[i][j];
	}}
	printf("\n");
	while(flag)
	{
	flag=0;
	for(i=0;i<n;i++){
	int c=0;
	for(j=0;j<r;j++){
	if((finish[i]==0)&&(need[i][j]<=avail[j]))
	{
	c++;
	if(c==r)
	{
	for(k=0;k<r;k++){
```

```
avail[k]+=alloc[i][j];
finish[i]=1;
flag=1;
}
printf("P%d->",i);
if(finish[i]==1)
{
i=n;
}}}}}}
for(i=0;i<n;i++){
if(finish[i]==1)
{
c1++;
}
else
{
printf("P%d->",i);
}
}
if(c1==n)
{
printf("\n The system is in safe state");
}
else
{
printf("\n Process are in dead lock");
printf("\n System is in unsafe state");
}
}
```

OUTPUT:



CONCLUSION:

In conclusion, the Banker's Algorithm is a vital resource allocation and deadlock avoidance technique used in operating systems. Its primary goal is to ensure the safe allocation of resources to processes while preventing deadlock, a situation where processes are unable to proceed due to circular dependency on resources. By dynamically analyzing the current state of the system and potential future resource requests, the algorithm optimizes resource utilization without compromising system safety. Through careful management of available resources, process requests, and system state, the Banker's Algorithm effectively maintains a balance between resource allocation and deadlock avoidance, contributing to the stability and efficiency of operating systems.