Name - Harshit

MIS – 112316018

OS lab

Assignment 8

To write a C program to implement banker's algorithm for deadlock avoidance.

Code:  
#include <stdio.h>

#include <stdbool.h>

#define MAX\_PROCESSES 10

#define MAX\_RESOURCES 10

int main() {

    int processes, resources;

    int allocation[MAX\_PROCESSES][MAX\_RESOURCES], max[MAX\_PROCESSES][MAX\_RESOURCES], need[MAX\_PROCESSES][MAX\_RESOURCES];

    int available[MAX\_RESOURCES], work[MAX\_RESOURCES];

    bool finish[MAX\_PROCESSES] = {false};

    int safeSequence[MAX\_PROCESSES], count = 0;

    printf("Enter the number of processes: ");

    scanf("%d", &processes);

    printf("Enter the number of resources: ");

    scanf("%d", &resources);

    printf("Enter the Allocation Matrix:\n");

    for (int i = 0; i < processes; i++) {

        for (int j = 0; j < resources; j++) {

            scanf("%d", &allocation[i][j]);

        }

    }

    printf("Enter the Max Matrix:\n");

    for (int i = 0; i < processes; i++) {

        for (int j = 0; j < resources; j++) {

            scanf("%d", &max[i][j]);

            need[i][j] = max[i][j] - allocation[i][j];

        }

    }

    printf("Enter Available Resources:\n");

    for (int j = 0; j < resources; j++) {

        scanf("%d", &available[j]);

        work[j] = available[j];

    }

    while (count < processes) {

        bool found = false;

        for (int i = 0; i < processes; i++) {

            if (!finish[i]) {

                bool canAllocate = true;

                for (int j = 0; j < resources; j++) {

                    if (need[i][j] > work[j]) {

                        canAllocate = false;

                        break;

                    }

                }

                if (canAllocate) {

                    for (int j = 0; j < resources; j++) {

                        work[j] += allocation[i][j];

                    }

                    safeSequence[count++] = i;

                    finish[i] = true;

                    found = true;

                }

            }

        }

        if (!found) {

            printf("The system is in an unsafe state! Deadlock may occur.\n");

            return 1;

        }

    }

    printf("The system is in a safe state.\nSafe Sequence: ");

    for (int i = 0; i < processes; i++) {

        printf("P%d ", safeSequence[i]);

    }

    printf("\n");

    return 0;

}

Output:  
