PRACTICAL NO: 05

CODE:

#include <stdio.h>

#include <stdbool.h>

int main() {

int n, m, i, j, k;

// n = number of processes, m = number of resource types

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

scanf("%d", &n);

printf("Enter the number of resource types: ");

scanf("%d", &m);

int alloc[n][m]; // Allocation matrix

int max[n][m]; // Maximum demand matrix

int avail[m]; // Available resources

int need[n][m]; // Need matrix

int work[m]; // Work vector (used in safety check)

bool finish[n]; // Finish array to mark processes

int safeSeq[n]; // Safe sequence of processes

// Input allocation matrix

printf("\nEnter the allocation matrix (row-wise):\n");

for (i = 0; i < n; i++) {

for (j = 0; j < m; j++) {

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

}

}

// Input maximum demand matrix

printf("\nEnter the maximum demand matrix (row-wise):\n");

for (i = 0; i < n; i++) {

for (j = 0; j < m; j++) {

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

}

}

// Input available resources

printf("\nEnter the available resources: ");

for (i = 0; i < m; i++) {

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

}

// Calculate the need matrix (Need = Max - Alloc)

for (i = 0; i < n; i++) {

for (j = 0; j < m; j++) {

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

}

}

// Copy available resources to the work vector

for (i = 0; i < m; i++) {

work[i] = avail[i];

}

// Initialize the finish array to false (indicating no processes are finished)

for (i = 0; i < n; i++) {

finish[i] = false;

}

// Banker's algorithm - Safety check

int count = 0;

while (count < n) {

bool found = false;

for (i = 0; i < n; i++) {

if (!finish[i]) {

// Check if need[i] <= work for all resource types

for (j = 0; j < m; j++) {

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

break;

}

}

// If all needs are satisfied for process i

if (j == m) {

// Add allocated resources of process i to work

for (k = 0; k < m; k++) {

work[k] += alloc[i][k];

}

safeSeq[count++] = i;

finish[i] = true;

found = true;

}

}

}

// If no process was found in this iteration, the system is unsafe

if (!found) {

printf("System is not in a safe state!\n");

return 1;

}

}

// If system is safe, print the safe sequence

printf("\nSystem is in a safe state.\nSafe sequence is: ");

for (i = 0; i < n; i++) {

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

}

printf("\n");

return 0;

}

Output:

