Program 5. Develop a C program to simulate Bankers Algorithm for DeadLock Avoidance.

#include <stdio.h>

#define MAX\_PROCESS 10

#define MAX\_RESOURCE 10

int main() {

int n, m, i, j, k;

int allocation[MAX\_PROCESS][MAX\_RESOURCE],

max[MAX\_PROCESS][MAX\_RESOURCE], available[MAX\_RESOURCE];

int need[MAX\_PROCESS][MAX\_RESOURCE], finish[MAX\_PROCESS], safeSeq[MAX\_PROCESS], work[MAX\_RESOURCE];

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

scanf("%d", &n);

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

scanf("%d", &m);

printf("Enter the allocation matrix:\n");

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

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

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

}

}

printf("Enter the maximum matrix:\n");

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

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

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

}

}

printf("Enter the available resources:\n"); for (i = 0; i < m; ++i) {

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

}

// Initialize finish array

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

finish[i] = 0;

}

// Calculate need matrix

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

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

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

}

// Initialize work array with available resources for (i = 0; i < m; ++i) {

work[i] = available[i];

}

int count = 0;

while (count < n) {

int found = 0;

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

if (finish[i] == 0) {

int flag = 1;

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

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

flag = 0;

break;

}

}

if (flag) {

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

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

}

safeSeq[count++] = i;

finish[i] = 1;

found = 1;

}

}

}

if (!found) {

printf("System is not in a safe state!\n"); return 0;

}

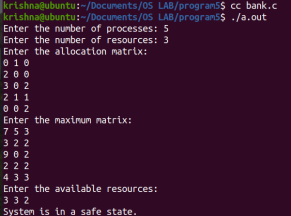
}

printf("System is in a safe state.\n ");

return 0;

}

**Output:**

****