**EXPERIMENT – 17**

17. Illustrate the deadlock avoidance concept by simulating Banker’s algorithm with C.

#include <stdio.h>

#include <stdbool.h>

#define MAX\_PROCESSES 10

#define MAX\_RESOURCES 10

int main() {

int n, m;

int allocation[MAX\_PROCESSES][MAX\_RESOURCES];

int max[MAX\_PROCESSES][MAX\_RESOURCES];

int need[MAX\_PROCESSES][MAX\_RESOURCES];

int available[MAX\_RESOURCES];

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

printf("Process %d: ", i);

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

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

}

}

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

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

printf("Process %d: ", i);

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

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

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

}

}

printf("Enter the Available Resources: ");

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

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

}

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

int safeSeq[MAX\_PROCESSES];

int count = 0;

for (int k = 0; k < n; k++) {

bool found = false;

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

if (!finish[i]) {

bool canAllocate = true;

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

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

canAllocate = false;

break;

}

}

if (canAllocate) {

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

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

}

safeSeq[count++] = i;

finish[i] = true;

found = true;

}

}

}

if (!found) break;

}

if (count == n) {

printf("\nSystem is in a SAFE state.\nSafe Sequence: ");

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

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

}

printf("\n");

} else {

printf("\nSystem is in an UNSAFE state. Deadlock may occur.\n");

}

return 0;

}

SAMPLE INPUT:

Enter the number of processes: 5

Enter the number of resources: 3

Enter the Allocation Matrix:

Process 0: 0 1 0

Process 1: 2 0 0

Process 2: 3 0 2

Process 3: 2 1 1

Process 4: 0 0 2

Enter the Max Matrix:

Process 0: 7 5 3

Process 1: 3 2 2

Process 2: 9 0 2

Process 3: 2 2 2

Process 4: 4 3 3

Enter the Available Resources: 3 3 2

SAMPLE OUTPUT:

System is in a SAFE state.

Safe Sequence: P1 P3 P4 P0 P2