Deadlock detection and Avoidance

#include <stdio.h>

#include <stdbool.h>

#define MAX\_P 10

#define MAX\_R 10

int p, r;

int allocation[MAX\_P][MAX\_R];

int max[MAX\_P][MAX\_R];

int available[MAX\_R];

int need[MAX\_P][MAX\_R];

int request[MAX\_P][MAX\_R];

void inputMatrices() {

printf("Enter number of processes: ");

scanf("%d", &p);

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

scanf("%d", &r);

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

for (int i = 0; i < p; i++)

for (int j = 0; j < r; j++)

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

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

for (int i = 0; i < p; i++)

for (int j = 0; j < r; j++)

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

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

for (int i = 0; i < r; i++)

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

printf("Enter Request Matrix (for Deadlock Detection):\n");

for (int i = 0; i < p; i++)

for (int j = 0; j < r; j++)

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

}

void calculateNeed() {

for (int i = 0; i < p; i++)

for (int j = 0; j < r; j++)

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

}

bool isSafe() {

int work[MAX\_R];

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

for (int i = 0; i < r; i++)

work[i] = available[i];

int count = 0;

while (count < p) {

bool found = false;

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

if (!finish[i]) {

int j;

for (j = 0; j < r; j++)

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

break;

if (j == r) {

for (int k = 0; k < r; k++)

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

finish[i] = true;

found = true;

count++;

}

}

}

if (!found)

return false;

}

return true;

}

bool detectDeadlock() {

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

int work[MAX\_R];

for (int i = 0; i < r; i++)

work[i] = available[i];

bool changed;

do {

changed = false;

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

if (!finish[i]) {

int j;

for (j = 0; j < r; j++)

if (request[i][j] > work[j])

break;

if (j == r) {

for (int k = 0; k < r; k++)

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

finish[i] = true;

changed = true;

}

}

}

} while (changed);

for (int i = 0; i < p; i++)

if (!finish[i])

return true;

return false;

}

int main() {

inputMatrices();

calculateNeed();

printf("\n--- Deadlock Avoidance (Banker's Algorithm) ---\n");

if (isSafe())

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

else

printf("System is NOT in a safe state.\n");

printf("\n--- Deadlock Detection ---\n");

if (detectDeadlock())

printf("Deadlock detected in the system.\n");

else

printf("No deadlock detected.\n");

return 0;

}

