BANKERS ALGORITHM FOR DEADLOCK AVOIDANCE

#include<stdio.h>

#include<conio.h>

int max[100][100];

int alloc[100][100];

int need[100][100];

int avail[100];

int n,r;

void input();

void show();

void cal();

int main() {

printf("\*\*\*\*\*\*\*\*\*\* Banker's Algorithm \*\*\*\*\*\*\*\*\*\*\*\*\n");

input();

show();

cal();

getch();

return 0;

}

void input() {

int i,j;

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

scanf("%d", &n);

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

scanf("%d", &r);

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

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

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

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

}

}

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

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

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

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

}

}

printf("Enter the available Resources\n");

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

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

}

}

void show() {

int i, j;

printf("\nProcess\tAllocation\tMax\tAvailable\n");

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

printf("P%d\t", i+1);

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

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

}

printf("\t");

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

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

}

printf("\t");

if(i == 0) {

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

printf("%d ", avail[j]);

}

}

printf("\n");

}

}

void cal() {

int finish[100], need[100][100];

int flag = 1, k, c1 = 0;

int safe[100];

int i, j;

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

finish[i] = 0;

}

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

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

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

}

}

printf("\nSafe Sequence: ");

while(flag) {

flag = 0;

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

int c = 0;

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

if((finish[i] == 0) && (need[i][j] <= avail[j])) {

c++;

if(c == r) {

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

avail[k] += alloc[i][j];

finish[i] = 1;

flag = 1;

}

printf("P%d -> ", i);

if(finish[i] == 1) {

i = n;

}

}

}

}

}

}

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

if(finish[i] == 1) {

c1++;

} else {

printf("P%d -> ", i);

}

}

if(c1 == n) {

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

} else {

printf("\n\nProcesses are in a deadlock.\n");

printf("System is in an unsafe state.\n");

}

}

OUTPUT:

