//Programmer Name: Sharvil Prabhudesai 20co41

//Program title : Banker’s Algorithm

#include<stdio.h>

#include<string.h>

int max[10][10],allocation[10][10],need[10][10],available[10],safe\_state[20];

int s=0; //safe\_state current index position

int n, m,i,j,k;

int work[10],finish[10],request[10];

int flag;

void input() {

int i, j;

printf("Enter the no of Processes : ");

scanf("%d", &n);

printf("Enter the no of resources instances : ");

scanf("%d", &m);

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

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

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

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

}

}

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 available Resources\n");

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

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

}

//find need matrix

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

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

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

}

}

}

void display() {

int i, j;

printf("Process\t\t Allocation\tMax\t\t Need\t\tAvailable");

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

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

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

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

}

printf("\t\t");

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

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

}

printf("\t\t");

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

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

}

printf("\t\t ");

if (i == 0) {

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

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

}

}

}

void safety\_sequence() {

//Making finish[i] = false , ie 0 for all i=0,1,...,n-1

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

finish[i]=0;

}

//Initialising Work =available

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

work[i]=available[i];

}

for(j=0;j<3;j++){

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

flag=1;

if(finish[i]==0){

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

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

flag=0;

break;

}

}

if(flag==1){

finish[i]=1;

safe\_state[s] = i+1;

s++;

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

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

}

}

}

}

}

flag=1; // flag used to check if all finish of i == true

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

if(finish[i]==0){

flag=0;

break;

}

}

if(flag==0){

printf("\n\nSystem is in Unsafe state !");

}

else{

printf("\n\nSystem is in Safe state !");

printf("\nThe Safe Sequence is :");

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

printf(" %d,",safe\_state[i]);

}

}

}

void resource\_request() {

flag=1;

printf("\nEnter the process number that makes the request : ");

scanf("%d",&i);

i=i-1; //decrementing since array index begins from zero

printf("Enter the request vector : ");

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

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

}

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

if(request[k]>need[i][k]){

flag=0;

break;

}

}

if (flag==0) {

printf("\n\nError Condition , request exceeded its max claim!");

return;

}

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

if(request[k]>available[k]){

flag=0;

break;

}

}

if (flag==0) {

printf("\n\nThe Process must wait ,Resources are not available!");

return;

}

printf("Resource from process can be granted immediately");

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

available[k]=available[k]-request[k];

need[i][k]=need[i][k]-request[k];

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

}

printf("\nAfter Granting Resources :\n");

display();

safety\_sequence();

}

int main() {

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

input();

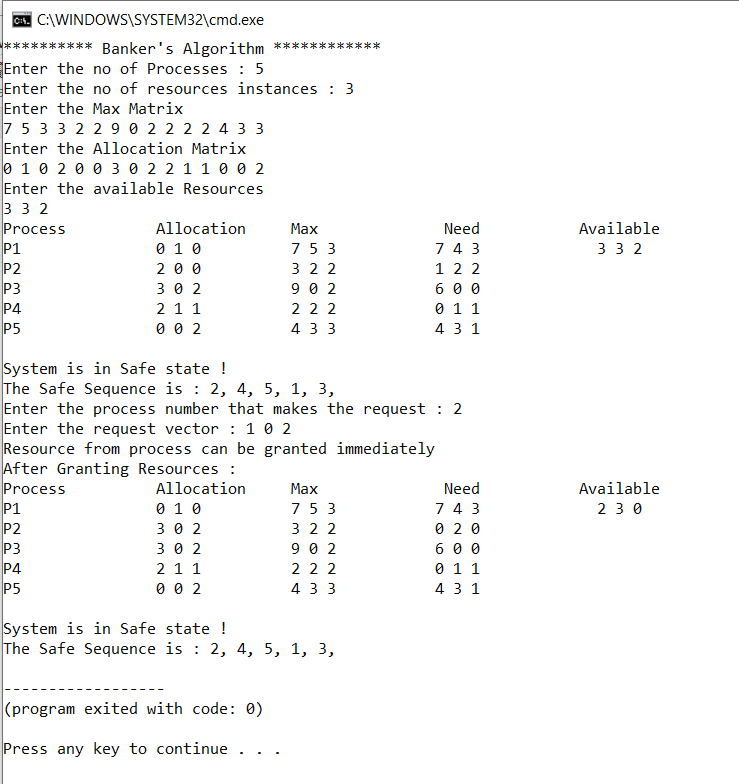
display();

safety\_sequence();

resource\_request();

return 0;

}



//Programmer Name: Sharvil Prabhudesai 20co41

//Program title : Deadlock Detection

#include<stdio.h>

int allocation[10][10],request[10][10],available[10];

int work[10],finish[10];

int i,j,n,m,k;

void input(){

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

scanf("%d",&n);

printf("Enter the Number of Resource Instances : ");

scanf("%d",&m);

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

//Allocation matrix input

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

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

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

}

}

//Request matrixc input

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

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

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

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

}

}

//Available Vector Input

printf("Enter the Available Vector : \n");

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

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

}

}

void display() {

int i, j;

printf("Process\t\t Allocation\tRequest\t\t Available");

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

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

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

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

}

printf("\t\t");

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

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

}

printf("\t\t ");

if (i == 0) {

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

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

}

}

}

void deadlock\_detection(){

int flag;

//Making finish[i] = false , ie 0 for all i=0,1,...,n-1

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

finish[i]=0;

}

//Initialising Work =available

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

work[i]=available[i];

}

for(j=0;j<3;j++){

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

flag=1;

if(finish[i]==0){

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

if(request[i][k]>work[k]){

flag=0;

break;

}

}

if(flag==1){

finish[i]=1;

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

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

}

}

}

}

}

flag=1; // flag used to check if all finish of i == true

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

if(finish[i]==0)

{

flag=0;

}

}

if(flag==0){

printf("\n\nSystem is in Deadlock state !");

}

else{

printf("\n\nSystem is not in Deadlock state !");

}

}

int main(){

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

input();

display();

deadlock\_detection();

return 0;

}

