BANKERS ALGORITHM

#include <stdio.h>

#include <stdbool.h>

#define n 5 // Number of processes

#define m 3 // Number of resource types

int main() {

int i, j, k;

int available[m];

int allocation[n][m];

int max[n][m];

bool finish[n] = {false};

int safeSeq[n];

int work[m];

// Input Allocation Matrix

printf("Enter the resource Allocation for processes:\n\Resource A B C respectively :\n");

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

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

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

}

}

// Input Max Demand Matrix

printf("Enter the max demand for resources from processes :\n\Resource A B C respectively :\n");

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

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

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

}

}

// Input Available Resources Vector

printf("Enter the Available Resources Vector:\nResource A B C respectively :\n");

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

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

}

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

work[i] = available[i];

}

int count = 0;

while (count < n) {

bool found = false;

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

if (!finish[i]) {

bool canExecute = true;

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

if (max[i][j] - allocation[i][j] > work[j]) {

canExecute = false;

break;

}

}

if (canExecute) {

finish[i] = true;

safeSeq[count++] = i;

found = true;

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

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

}

break;

}

}

}

if (!found) {

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

return -1;

}

}

printf("System is in a safe state.\nSafe Sequence: ");

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

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

}

printf("\n");

return 0;

OUTPUT:

}