Banker’s algorithm

#include <stdio.h>

int main() {

int n, m, i, j, k;

printf("Enter number of processes: ");

scanf("%d", &n);

printf("Enter number of resources: ");

scanf("%d", &m);

int alloc[n][m], max[n][m], avail[m];

int need[n][m], finish[n], safeSeq[n];

printf("Enter allocation matrix:\n");

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

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

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

printf("Enter max matrix:\n");

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

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

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

printf("Enter available resources:\n");

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

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

// Calculate need matrix

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

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

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

int count = 0;

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

finish[i] = 0;

while (count < n) {

int found = 0;

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

if (!finish[i]) {

int canAllocate = 1;

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

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

canAllocate = 0;

break;

}

}

if (canAllocate) {

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

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

safeSeq[count++] = i;

finish[i] = 1;

found = 1;

}

}

}

if (!found) {

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

return 1;

}

}

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

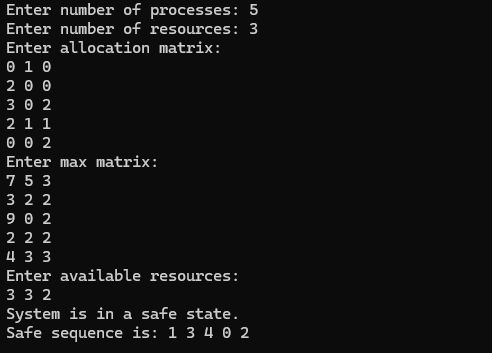
for (i = 0; i < n; i++)

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

printf("\n");

return 0;

}



**Deadlock detection**

#include <stdio.h>

int main() {

int n, m, i, j, k;

printf("Enter number of processes: ");

scanf("%d", &n);

printf("Enter number of resources: ");

scanf("%d", &m);

int alloc[n][m], request[n][m], avail[m], work[m], finish[n];

printf("Enter allocation matrix:\n");

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

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

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

printf("Enter request matrix:\n");

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

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

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

printf("Enter available resources:\n");

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

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

work[i] = avail[i];

}

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

int allocated = 0;

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

if (alloc[i][j] != 0)

allocated = 1;

finish[i] = (allocated == 0) ? 1 : 0;

}

int changed;

do {

changed = 0;

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

if (!finish[i]) {

int canProceed = 1;

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

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

canProceed = 0;

break;

}

if (canProceed) {

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

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

finish[i] = 1;

changed = 1;

}

}

}

} while (changed);

int deadlock = 0;

printf("Processes in deadlock: ");

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

if (!finish[i]) {

printf("%d ", i);

deadlock = 1;

}

}

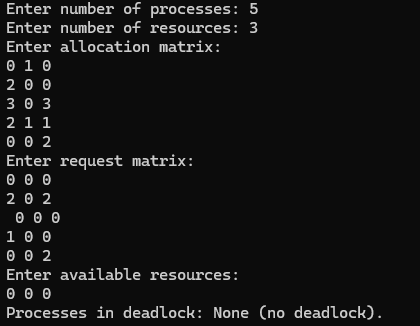
if (!deadlock)

printf("None (no deadlock).");

printf("\n");

return 0;

}

****