**BANKER’S ALGORITHM**

#include <stdio.h>

#define P 5 // Number of processes

#define R 3 // Number of resources

int main() {

int i, j, k;

// Example input

int available[R] = {3, 3, 2}; // Available instances of each resource

int max[P][R] = {

{7, 5, 3},

{3, 2, 2},

{9, 0, 2},

{2, 2, 2},

{4, 3, 3}

};

int allocation[P][R] = {

{0, 1, 0},

{2, 0, 0},

{3, 0, 2},

{2, 1, 1},

{0, 0, 2}

};

int need[P][R];

int finish[P] = {0};

int safeSeq[P];

int count = 0;

// Calculate need matrix

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

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

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

}

}

// Banker's algorithm to find safe sequence

while (count < P) {

int found = 0;

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

if (finish[i] == 0) {

int canAllocate = 1;

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

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

canAllocate = 0;

break;

}

}

if (canAllocate) {

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

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

}

safeSeq[count++] = i;

finish[i] = 1;

found = 1;

}

}

}

if (!found) {

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

return 1;

}

}

// Print safe sequence

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

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

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

}

printf("\n");

return 0;

}