#include <stdio.h>

// Define the number of processes and resources

#define NUM\_PROCESSES 5

#define NUM\_RESOURCES 3

// Available resources

int available[NUM\_RESOURCES] = {3, 3, 2};

// Maximum demand of each process

int max\_demand[NUM\_PROCESSES][NUM\_RESOURCES] = {

{7, 5, 3},

{3, 2, 2},

{9, 0, 2},

{2, 2, 2},

{4, 3, 3}

};

// Allocated resources to processes

int allocation[NUM\_PROCESSES][NUM\_RESOURCES] = {

{0, 1, 0},

{2, 0, 0},

{3, 0, 2},

{2, 1, 1},

{0, 0, 2}

};

// Need resources for each process

int need[NUM\_PROCESSES][NUM\_RESOURCES];

// Function to check if the system is in a safe state

int isSafeState(int process, int request[]) {

// Step 1: Check if the request is less than or equal to the need

for (int i = 0; i < NUM\_RESOURCES; i++) {

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

return 0; // Request exceeds need

}

}

// Step 2: Check if the request is less than or equal to the available resources

for (int i = 0; i < NUM\_RESOURCES; i++) {

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

return 0; // Request exceeds available resources

}

}

// Try to allocate the resources and see if it remains safe

for (int i = 0; i < NUM\_RESOURCES; i++) {

available[i] -= request[i];

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

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

}

int finish[NUM\_PROCESSES] = {0};

int work[NUM\_RESOURCES];

for (int i = 0; i < NUM\_RESOURCES; i++) {

work[i] = available[i];

}

int count = 0;

while (count < NUM\_PROCESSES) {

int found = 0;

for (int i = 0; i < NUM\_PROCESSES; i++) {

if (finish[i] == 0) {

int j;

for (j = 0; j < NUM\_RESOURCES; j++) {

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

break;

}

}

if (j == NUM\_RESOURCES) {

for (j = 0; j < NUM\_RESOURCES; j++) {

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

}

finish[i] = 1;

found = 1;

count++;

}

}

}

if (!found) {

for (int i = 0; i < NUM\_RESOURCES; i++) {

available[i] += request[i];

allocation[process][i] -= request[i];

need[process][i] += request[i];

}

return 0; // System is not in a safe state

}

}

// If we reach here, the system is in a safe state

return 1;

}

// Function to allocate resources to a process

void allocateResources(int process, int request[]) {

if (isSafeState(process, request)) {

printf("Resource request for Process %d granted.\n", process);

} else {

printf("Resource request for Process %d denied (would lead to deadlock).\n", process);

}

}

int main() {

// Initialize the 'need' matrix

for (int i = 0; i < NUM\_PROCESSES; i++) {

for (int j = 0; j < NUM\_RESOURCES; j++) {

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

}

}

// Test resource allocation requests

int process = 1;

int request[NUM\_RESOURCES] = {1, 0, 2};

allocateResources(process, request);

process = 3;

int request2[NUM\_RESOURCES] = {0, 2, 0};

allocateResources(process, request2);

return 0;

}