#include <stdio.h>

#include <stdbool.h>

#define MAX 10

#define RESOURCE\_TYPES 10

bool request\_granted(int n, int m, int request[], int available[], int allocation[][RESOURCE\_TYPES], int need[][RESOURCE\_TYPES], int process\_id) {

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

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

return false;

}

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

return false;

}

}

return true;

}

void update\_resources(int n, int m, int request[], int available[], int allocation[][RESOURCE\_TYPES], int need[][RESOURCE\_TYPES], int process\_id) {

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

available[i] -= request[i];

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

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

}

}

bool safety\_algorithm(int n, int m, int available[], int allocation[][RESOURCE\_TYPES], int need[][RESOURCE\_TYPES], int safe\_sequence[]) {

int work[RESOURCE\_TYPES];

bool finish[MAX] = {false};

int count = 0;

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

work[i] = available[i];

}

while (count < n) {

bool progress\_made = false;

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

if (!finish[i]) {

bool can\_finish = true;

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

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

can\_finish = false;

break;

}

}

if (can\_finish) {

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

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

}

safe\_sequence[count++] = i;

finish[i] = true;

progress\_made = true;

break;

}

}

}

if (!progress\_made) {

return false;

}

}

return true;

}

int main() {

printf("Yash Sigchi-23BAI1242:\n\n");

int n, m;

printf("Enter the number of processes(23BAI1242): ");

scanf("%d", &n);

printf("Enter the number of resource types(23BAI1242): ");

scanf("%d", &m);

int allocation[MAX][RESOURCE\_TYPES], maximum[MAX][RESOURCE\_TYPES], need[MAX][RESOURCE\_TYPES], available[RESOURCE\_TYPES];

int safe\_sequence[MAX];

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

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

printf("Process %d allocation: ", i);

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

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

}

}

printf("Enter the maximum matrix:\n");

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

printf("Process %d maximum: ", i);

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

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

}

}

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

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

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

}

}

printf("Enter the available resources: ");

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

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

}

bool is\_safe = safety\_algorithm(n, m, available, allocation, need, safe\_sequence);

if (is\_safe) {

printf("The system is in a safe state.\n(23BAI1242)\n");

printf("Safe sequence: ");

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

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

}

printf("\n");

} else {

printf("The system is in an unsafe state.\n(23BAI1242)");

}

int process\_id;

printf("Enter the process id requesting resources: ");

scanf("%d", &process\_id);

int request[RESOURCE\_TYPES];

printf("Enter the resource request for process %d: ", process\_id);

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

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

}

if (request\_granted(n, m, request, available, allocation, need, process\_id)) {

printf("Request can be granted.\n");

update\_resources(n, m, request, available, allocation, need, process\_id);

is\_safe = safety\_algorithm(n, m, available, allocation, need, safe\_sequence);

if (is\_safe) {

printf("The system is still in a safe state after the request.\n");

} else {

printf("The system is in an unsafe state after the request. Rolling back.\n");

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

available[i] += request[i];

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

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

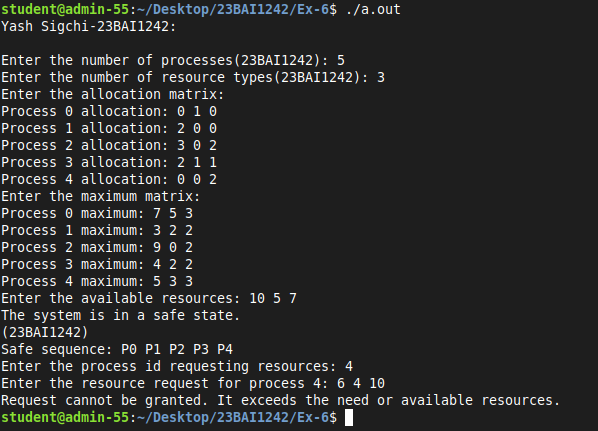
}

}

} else {

printf("Request cannot be granted. It exceeds the need or available resources.\n");

}

return 0;

}

