Banker’s algorithm  
#include <stdio.h>

#include <stdbool.h>

#define P 5

#define R 3

int main() {

int alloc[P][R] = {{0, 1, 0}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2}};

int max[P][R] = {{7, 5, 3}, {3, 2, 2}, {9, 0, 2}, {2, 2, 2}, {4, 3, 3}};

int avail[R] = {3, 3, 2};

int need[P][R];

int finish[P] = {0};

int safeSeq[P];

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

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

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

int count = 0;

while (count < P) {

bool found = false;

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

if (!finish[i]) {

int j;

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

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

break;

if (j == R) {

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

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

safeSeq[count++] = i;

finish[i] = 1;

found = true;

}

}

}

if (!found) {

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

return 1;

}

}

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

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

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

printf("\n");

return 0;

}