1.Consider a system with 4 processes and 3 resources with the given resource matrices.

Claim matrix Allocation matrix

3 2 2 1 0 0

6 1 3 6 1 2

3 1 4 2 1 1

4 2 2 0 0 2

The resource vector is [9,3,6]. Write a C program to determine if the system is in safe or unsafe state.

Program:

#include <stdio.h>

#include <stdbool.h>

#define NUM\_PROCESSES 4

#define NUM\_RESOURCES 3

int main() {

// initialize the resource matrices

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

{3, 2, 2},

{6, 1, 3},

{3, 1, 4},

{4, 2, 2}

};

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

{1, 0, 0},

{6, 1, 2},

{2, 1, 1},

{0, 0, 2}

};

int available[NUM\_RESOURCES] = {9, 3, 6};

// initialize the work and finish arrays

int work[NUM\_RESOURCES];

bool finish[NUM\_PROCESSES];

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

work[i] = available[i];

}

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

finish[i] = false;

}

// find a process that can run

bool found = true;

while (found) {

found = false;

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

if (!finish[i]) {

bool can\_run = true;

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

if (claim[i][j] - allocation[i][j] > work[j]) {

can\_run = false;

break;

}

}

if (can\_run) {

found = true;

finish[i] = true;

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

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

}

}

}

}

}

// check if all processes have finished

bool safe = true;

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

if (!finish[i]) {

safe = false;

break;

}

}

// print the result

if (safe) {

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

} else {

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

}

return 0;

}

Output:

