```
#include <stdio.h>
#include <stdbool.h>

#define MAX_PROCESSES 100
#define MAX_RESOURCES 100

int available[MAX_RESOURCES];
int maximum[MAX_PROCESSES][MAX_RESOURCES];
int allocation[MAX_PROCESSES][MAX_RESOURCES];
int need[MAX_PROCESSES][MAX_RESOURCES];
bool finish[MAX_PROCESSES];
int num_processes, num_resources;

bool is_safe_state();

int main()
{
   printf("Enter the number of processes: ");
   scanf("%d", &num_processes);
   printf("Enter the number of resources: ");
   scanf("%d", &num_resources);

   printf("Enter the available resources:\n");
   for (int i = 0; i < num_resources; i++) {
      scanf("%d", &available[i]);
   }

   printf("Enter the maximum resource allocation for each process:\n");
   for (int i = 0; i < num_processes; i++) {
      printf("Process %d: ", i);
      for (int j = 0; j < num_resources; j++) {
```

```
            scanf("%d", &maximum[i][j]);
            need[i][j] = maximum[i][j];
        }
    }

    printf("Enter the current resource allocation for each process:\n");
    for (int i = 0; i < num_processes; i++) {
        printf("Process %d: ", i);
        for (int j = 0; j < num_resources; j++) {
            scanf("%d", &allocation[i][j]);
            need[i][j] -= allocation[i][j];
            available[j] -= allocation[i][j];
        }
    }

    if (is_safe_state()) {
        printf("The system is in a safe state.\n");
    } else {
        printf("The system is in an unsafe state.\n");
    }

    return 0;
}

bool is_safe_state()
{
    int work[MAX_RESOURCES];
    bool found;

    for (int i = 0; i < num_resources; i++) {
        work[i] = available[i];
```

```
}

for (int i = 0; i < num_processes; i++) {
    finish[i] = false;
}

int count = 0;
while (count < num_processes) {
    found = false;
    for (int i = 0; i < num_processes; i++) {
        if (!finish[i]) {
            int j;
            for (j = 0; j < num_resources; j++) {
                if (need[i][j] > work[j]) {
                    break;
                }
            }
            if (j == num_resources) {
                for (int k = 0; k < num_resources; k++) {
                    work[k] += allocation[i][k];
                }
                finish[i] = true;
                found = true;
                count++;
            }
        }
    }
    if (!found) {
        break;
    }
}
```

```
    if (count == num_processes) {
        return true;
    } else {
        return false;
    }
}
```