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

#define MAX 10
#define RESOURCE_TYPES 3

void calculate_need(int need[MAX][RESOURCE_TYPES], int max[MAX][RESOURCE_TYPES],
                    int allot[MAX][RESOURCE_TYPES], int n) {
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < RESOURCE_TYPES; j++) {
      need[i][j] = max[i][j] - allot[i][j];
    }
  }
}

bool is_safe(int processes[], int n, int m, int available[],
             int max[][RESOURCE_TYPES], int allot[][RESOURCE_TYPES],
             int need[MAX][RESOURCE_TYPES]) {
  int work[RESOURCE_TYPES];
  bool finish[MAX];

  for (int i = 0; i < m; i++) {
    work[i] = available[i];
  }

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

  int safeSeq[MAX];
  int count = 0;

  while (count < n) {
    bool found = false;
    for (int p = 0; p < n; p++) {
      if (!finish[p]) {
        int j;
        for (j = 0; j < m; j++) {
          if (need[p][j] > work[j]) {
            break;
          }
        }
        if (j == m) {
          for (int k = 0; k < m; k++) {
            work[k] += allot[p][k];
          }
          safeSeq[count++] = p;
          finish[p] = true;
          found = true;
        }
      }
    }
    if (!found) {
      return false;
    }
  }

  printf("System is in a safe state.\nSafe sequence is: ");
  for (int i = 0; i < n; i++) {
    printf("P%d ", safeSeq[i]);
  }
  printf("\n");
  return true;
}

void request_resources(int processes[], int n, int m, int available[],
                       int max[][RESOURCE_TYPES], int allot[][RESOURCE_TYPES],
                       int need[MAX][RESOURCE_TYPES]) {
  int request[MAX][RESOURCE_TYPES];
  int i, j, process_id;

  printf("Enter process number making request: ");
  scanf("%d", &process_id);

  if (process_id < 0 || process_id >= n) {
    printf("Invalid process number.\n");
```

```
    return;
  }

  printf("Enter request for resources:\n");
  for (i = 0; i < m; i++) {
    printf("Resource %d: ", i + 1);
    scanf("%d", &request[process_id][i]);
  }

  for (i = 0; i < m; i++) {
    if (request[process_id][i] > need[process_id][i]) {
      printf("Request exceeds maximum claim.\n");
      return;
    }
  }

  for (i = 0; i < m; i++) {
    if (request[process_id][i] > available[i]) {
      printf("Resources not available.\n");
      return;
    }
  }

  for (i = 0; i < m; i++) {
    available[i] -= request[process_id][i];
    allot[process_id][i] += request[process_id][i];
    need[process_id][i] -= request[process_id][i];
  }

  if (is_safe(processes, n, m, available, max, allot, need)) {
    printf("Request granted.\n");
  } else {
    printf("Request cannot be granted due to unsafe state. Rolling back.\n");

    for (i = 0; i < m; i++) {
      available[i] += request[process_id][i];
      allot[process_id][i] -= request[process_id][i];
      need[process_id][i] += request[process_id][i];
    }
  }
}

int main() {
  int n, m;
  int processes[MAX];
  int available[RESOURCE_TYPES];
  int max[MAX][RESOURCE_TYPES];
  int allot[MAX][RESOURCE_TYPES];
  int need[MAX][RESOURCE_TYPES];

  printf("Enter number of processes: ");
  scanf("%d", &n);
  printf("Enter number of resource types: ");
  scanf("%d", &m);

  printf("Enter the number of available instances for each resource:\n");
  for (int i = 0; i < m; i++) {
    printf("Resource %d: ", i + 1);
    scanf("%d", &available[i]);
  }

  printf("Enter maximum matrix:\n");
  for (int i = 0; i < n; i++) {
    processes[i] = i;
    printf("Process %d:\n", i + 1);
    for (int j = 0; j < m; j++) {
      printf("Maximum resource %d: ", j + 1);
      scanf("%d", &max[i][j]);
    }
  }

  printf("Enter allocation matrix:\n");
  for (int i = 0; i < n; i++) {
    printf("Process %d:\n", i + 1);
    for (int j = 0; j < m; j++) {
      printf("Allocated resource %d: ", j + 1);
      scanf("%d", &allot[i][j]);
```

```
    }
  }

  calculate_need(need, max, allot, n);

  request_resources(processes, n, m, available, max, allot, need);

  return 0;
}
```