```
#include <iostream>
#include <vector>
#include <set>

using namespace std;

const int MAX_RESOURCES = 3;

vector<int> available(MAX_RESOURCES);
vector<vector<int>> max_claim(2, vector<int>(MAX_RESOURCES));
vector<vector<int>> allocation(2, vector<int>(MAX_RESOURCES));
vector<set<int>> request(2);

bool is_safe() {
   vector<int> work = available;
   vector<bool> finish(2, false);

   // Process P0
   if (!finish[0] && request[0].empty()) {
      finish[0] = true;
      for (int j = 0; j < MAX_RESOURCES; ++j) {
         work[j] += allocation[0][j];
      }
   }

   // Process P1
   if (!finish[1] && request[1].empty()) {
      finish[1] = true;
      for (int j = 0; j < MAX_RESOURCES; ++j) {
         work[j] += allocation[1][j];
      }
```

```
    }

    // Check if all processes finished
    for (int i = 0; i < 2; ++i) {
        if (!finish[i]) {
            return false;
        }
    }

    return true;
}

void display_status() {
    cout << "Initial Status of the System:" << endl;
    cout << "Available Resources: ";
    for (int i = 0; i < MAX_RESOURCES; ++i) {
        cout << available[i] << " ";
    }
    cout << endl << "Max Claim Matrix   Allocation Matrix" << endl;
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < MAX_RESOURCES; ++j) {
            cout << max_claim[i][j] << " ";
        }
        cout << "      ";
        for (int j = 0; j < MAX_RESOURCES; ++j) {
            cout << allocation[i][j] << " ";
        }
        cout << endl;
    }
}
```

```
int main() {
    // Initialize available resources
    cout << "Enter the available resources:" << endl;
    for (int i = 0; i < MAX_RESOURCES; ++i) {
        cin >> available[i];
    }

    // Initialize maximum claim matrix for P0
    cout << "Enter the maximum claim matrix for P0:" << endl;
    for (int j = 0; j < MAX_RESOURCES; ++j) {
        cin >> max_claim[0][j];
    }

    // Initialize allocation matrix for P0
    cout << "Enter the allocation matrix for P0:" << endl;
    for (int j = 0; j < MAX_RESOURCES; ++j) {
        cin >> allocation[0][j];
    }

    // Initialize request matrix for P0
    cout << "Enter the request matrix for P0:" << endl;
    for (int j = 0; j < MAX_RESOURCES; ++j) {
        int req;
        cin >> req;
        request[0].insert(req);
    }

    // Initialize maximum claim matrix for P1
    cout << "Enter the maximum claim matrix for P1:" << endl;
    for (int j = 0; j < MAX_RESOURCES; ++j) {
        cin >> max_claim[1][j];
```

```
    }

    // Initialize allocation matrix for P1
    cout << "Enter the allocation matrix for P1:" << endl;
    for (int j = 0; j < MAX_RESOURCES; ++j) {
        cin >> allocation[1][j];
    }

    // Initialize request matrix for P1
    cout << "Enter the request matrix for P1:" << endl;
    for (int j = 0; j < MAX_RESOURCES; ++j) {
        int req;
        cin >> req;
        request[1].insert(req);
    }

    // Display the initial status
    display_status();

    // Check if the system is in a safe state
    if (is_safe()) {
        cout << "The system is in a safe state." << endl;
    } else {
        cout << "The system is in an unsafe state." << endl;
    }

    return 0;
}
```