// Implementation of Banker’s algorithm:

#include <bits/stdc++.h>

using namespace std;

int main()

{

    // resources A,B,C

    int n;

    cout << "Enter Number of proceses: ";

    cin >> n;

    cout << "Enter the total Number of resources of each type (A,B,C): ";

    int A, B, C;

    cin >> A >> B >> C;

    vector<vector<int>> allocate(n, vector<int>(3, 0));

    cout << "Enter Allocated resources: \n";

    for (int i = 0; i < n; i++)

    {

        cout << "Enter the allocated resources for P" << i + 1 << endl;

        for (int j = 0; j < 3; j++)

        {

            cin >> allocate[i][j];

        }

    }

    vector<vector<int>> maxreq(n, vector<int>(3, 0));

    cout << "Enter Maximum Required resources: \n";

    for (int i = 0; i < n; i++)

    {

        cout << "Enter the required resources for P" << i + 1 << endl;

        for (int j = 0; j < 3; j++)

        {

            cin >> maxreq[i][j];

        }

    }

    // vector<vector<int>>availres;

    cout << "Enter Currently Available resources of each type(A,B,C)\n";

    int ava, avb, avc;

    cin >> ava >> avb >> avc;

    vector<vector<int>> needed(n, vector<int>(3, 0));

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < 3; j++)

        {

            needed[i][j] = maxreq[i][j] - allocate[i][j];

        }

    }

    // needed matrix

    cout << "\n\n\nSr.NO\tA\tB\tC" << endl;

    cout << endl;

    for (int i = 0; i < n; i++)

    {

        cout << "P" << i + 1 << "\t";

        for (int j = 0; j < 3; j++)

        {

            cout << needed[i][j] << "\t";

        }

        cout << endl;

    }

    vector<int> processes;

    int completed = 0;

    while (completed != n)

    {

        for (int i = 0; i < n; i++)

        {

            if ((needed[i][0] <= ava && needed[i][0] != -1) && (needed[i][1] <= avb && needed[i][1] != -1) &&

                (needed[i][2] <= avc && needed[i][2] != -1))

            {

                completed++;

                processes.push\_back(i + 1);

                ava += allocate[i][0];

                avb += allocate[i][1];

                avc += allocate[i][2];

                needed[i][0] = -1;

                needed[i][1] = -1;

                needed[i][2] = -1;

            }

        }

    }

    cout << "\n\n\nCurrenty Available Resources: " << endl;

    cout << ava << " " << avb << " " << avc << endl;

    cout << "\n\nThe Order Of Completion of processes is: " << endl;

    for (int i = 0; i < n; i++)

    {

        cout << "P" << processes[i] << "->";

    }

    return 0;

}

// Output:

// hans@ryzu:/mnt/d/TE\_1sem/Labs/osl/05$ ./a.out

// Enter Number of proceses: 5

// Enter the total Number of resources of each type (A,B,C): 10 5 7

// Enter Allocated resources:

// Enter the allocated resources for P1

// 0 1 0

// Enter the allocated resources for P2

// 2 0 0

// Enter the allocated resources for P3

// 3 0 2

// Enter the allocated resources for P4

// 2 1 1

// Enter the allocated resources for P5

// 0 0 2

// Enter Maximum Required resources:

// Enter the required resources for P1

// 7 5 3

// Enter the required resources for P2

// 3 2 2

// Enter the required resources for P3

// 9 0 2

// Enter the required resources for P4

// 4 2 2

// Enter the required resources for P5

// 5 3 3

// Enter Currently Available resources of each type(A,B,C)

// 3 3 2

// Sr.NO A B C

// P1 7 4 3

// P2 1 2 2

// P3 6 0 0

// P4 2 1 1

// P5 5 3 1

// Currenty Available Resources:

// 10 5 7

// The Order Of Completion of processes is:

// P2->P4->P5->P1->P3