```
/**

NAME: Catrina Magdalena
DIVISION: SR3
GRADE: 11
SCHOOl: NCSC Grigore Moisil Brasov

**/

#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;

ifstream f("patolli.in");

struct marker{
    int number;
    int specrule = 0;
    int index, which;

}P[3], O[3];

int isOccupied[60];
int nrr, d;

bool crossed(int last_number, int actual_number)
{
    if(last_number <= 6 && actual_number >= 8)
        return 1;
    if(last_number <= 11 && actual_number >= 13)
        return 1;
    if(last_number <= 16 && actual_number >= 18)
        return 1;
    if(last_number <= 21 && actual_number >= 23)
        return 1;
    if(last_number <= 26 && actual_number >= 28)
        return 1;
    if(last_number <= 34 && actual_number >= 36)
        return 1;
    if(last_number <= 39 && actual_number >= 41)
        return 1;
    if(last_number <= 44 && actual_number >= 46)
        return 1;
    if(last_number <= 49 && actual_number >= 51)
        return 1;
    return 0;
}


bool isPrime(int x)
{
    if(x < 2 || x > 2 && x % 2 == 0)
        return 0;
```

```
    for(int d = 3; d*d <= x; d+=2)
        if(x%d == 0)
            return 0;
    return 1;
}

bool isPP(int x)
{
    float sq = sqrt(1.0*x);
    if((int)sq == sq)
        return 1;
    return 0;
}




int findSmallestMarker(marker A[3])
{
    int vm = 0;
    for(int i = 0; i < 3; ++i)
        if(A[i].number < A[vm].number)
            vm = i;
    return vm;
}

void moveM(int is, int step, marker A[3])
{
    isOccupied[52] = 0;
    if(A[is].number == 52)
        return;

    int wanted_pos = A[is].number + step;
    if(isOccupied[wanted_pos] == 1)
        return;
    if(wanted_pos > 52)
        return;

    if(isPrime(wanted_pos))
    {
        isOccupied[A[is].number] = 0;
        A[is].number = wanted_pos;
        isOccupied[A[is].number] = 1;
        for(int i = 1; i <= 6; ++i)
        {
            if(isOccupied[A[is].number + 1] == 1 || A[is].number + 1 >
52)
                return;
            isOccupied[A[is].number] = 0;
            A[is].number ++;
            isOccupied[A[is].number] = 1;
        }
        return;
```

```
    }
    if(isPP(wanted_pos) && wanted_pos > 4)
    {

        isOccupied[A[is].number] = 0;
        A[is].number = wanted_pos;
        isOccupied[A[is].number] = 1;
        for(int i = 1; i <= 6; ++i)
        {
            if(isOccupied[A[is].number - 1] == 1)
                return;
            isOccupied[A[is].number] = 0;
            A[is].number --;
            isOccupied[A[is].number] = 1;
        }
        return;
    }
    if(crossed(A[is].number, wanted_pos))
    {
        for(int nmb = wanted_pos; nmb > A[is].number; nmb--)
        {
            if(nmb % d == 0 && isOccupied[nmb] == 0)
            {
                isOccupied[A[is].number] = 0;
                A[is].number = nmb;
                isOccupied[A[is].number] = 1;
            }
        }
    }
    else{
        isOccupied[A[is].number] = 0;
        A[is].number = wanted_pos;
        isOccupied[A[is].number] = 1;
    }

}


void occupy()
{
    for(int i = 0; i < 3; ++i)
    {
        isOccupied[P[i].number] = 1;
        isOccupied[O[i].number] = 1;
    }
}




void read()
{
    f>>O[0].number>>O[1].number>>O[2].number;
```

```
    f>>P[0].number>>P[1].number>>P[2].number;
    occupy();
    f>>nrr;
    for(int i = 1; i <= nrr; ++i)
    {
        f>>d;
        int smallest;
        if(i%2 == 1) /// oponent
        {
            smallest = findSmallestMarker(O);
            moveM(smallest, d, O);
        }
        else{
            smallest = findSmallestMarker(P);
            moveM(smallest, d, P);
        }

    }
    int sumo = 0, sump = 0;
    for(int i = 0; i < 3; ++i)
    {
        if(P[i].number != 52)
            sump += P[i].number;
        if(O[i].number != 52)
            sumo += O[i].number;
    }
    cout<<sumo<<" "<<sump<<'\n';

}







int main()
{
    for(int acsl = 0; acsl < 5; ++acsl)
    {
        for(int i = 1; i <= 52; ++i)
            isOccupied[i] = 0;
        read();

    }
    return 0;
}
```