```
/*
    NAME: VELTAN Victor
    SCHOOL: CNI GRIGORE MOISIL
    GRADE: 12
    DIVISION: SR-3
*/

#include <bits/stdc++.h>

using namespace std;

ifstream f("patolli.in");

map<int,pair<int,int>>numToMat;
int mat[20][20], auxmat[20][20];
int pozAdversar[5], pozPlayer[5];
int n, dice;

void createNumToMat()
{
    numToMat[0]={1,1};
    for (int i=0; i<=11; ++i)
    {
        for (int j=0; j<=11; ++j)
            mat[i][j]=-1;
    }
    numToMat[1]= {1,5};
    mat[1][5]=0;
    numToMat[2]= {2,5};
    mat[2][5]=0;
    for (int i=3; i<=7; ++i)
    {
        numToMat[i]= {3,8-i};
        mat[3][8-i]=0;
    }
    for (int i=8; i<=12; ++i)
    {
        numToMat[i]= {4,i-7};
        mat[4][i-7]=0;
    }
    for (int i=13; i<=17; ++i)
    {
        numToMat[i]= {5,18-i};
        mat[5][18-i]=0;
    }
    for (int i=18; i<=22; ++i)
    {
        numToMat[i]= {6,i-17};
        mat[6][i-17]=0;
    }
    for (int i=23; i<=26; ++i)
    {
        numToMat[i]= {i-16,5};
        mat[i-16][5]=0;
```

```
    }
    for (int i=27; i<=30; ++i)
    {
        numToMat[i]= {37-i,6};
        mat[37-i][6]=0;
    }
    for (int i=31; i<=35; ++i)
    {
        numToMat[i]= {6,i-25};
        mat[6][i-25]=0;
    }
    for (int i=36; i<=40; ++i)
    {
        numToMat[i]= {5,46-i};
        mat[5][46-i]=0;
    }
    for (int i=41; i<=45; ++i)
    {
        numToMat[i]= {4,i-35};
        mat[4][i-35]=0;
    }
    for (int i=46; i<=50; ++i)
    {
        numToMat[i]= {3,56-i};
        mat[3][56-i]=0;
    }
    numToMat[51]= {2,6};
    mat[2][6]=0;
    numToMat[52]= {1,6};
    mat[1][6]=0;
    ///Tot ce este peste 53 sa dea o pozitie imposibila
    for (int i=53; i<=70; ++i)
    numToMat[i]={1,1};
    for (int i=1; i<=10; ++i)
    {
        for (int j=1; j<=10; ++j)
            auxmat[i][j]=mat[i][j];
    }
}

void resetMat()
{
    for (int i=1; i<=10; ++i)
    {
        for (int j=1; j<=10; ++j)
            mat[i][j]=auxmat[i][j];
    }
}

bool ePrim(int x)
{
    if (x%2==0)
        return false;
    for (int d=3; d*d<=x; d+=2)
```

```
    {
        if (x%d==0)
            return false;
    }
    return true;
}

void gasireUrmatoarePoz(int &poz, int ratie, int distanta, int pozActual)
{
    for (int i=1; i<=distanta; ++i)
    {
        if (poz+ratie!=pozActual &&
mat[numToMat[poz+ratie].first][numToMat[poz+ratie].second]!=0)
            return;
        else
            poz+=ratie;
    }
}

bool ePatrat(int x)
{
    if (x<=4)
        return false;
    for (int d=3; d<=7; ++d)
    {
        if (d*d==x)
            return true;
    }
    return false;
}

bool rule9(int pozUrmatoare, int pozActual)
{
//    if (abs(numToMat[pozActual].first-numToMat[pozUrmatoare].first)<1
|| abs(numToMat[pozActual].second-numToMat[pozUrmatoare].second)<1)
//        return false;
//    int x, y, rx, ry;
//    x=numToMat[pozActual].second;
//    y=numToMat[pozActual].first;
//    if (numToMat[pozActual].second>numToMat[pozUrmatoare].second)
//        rx=-1;
//    else
//        rx=1;
//    if (numToMat[pozActual].first>numToMat[pozUrmatoare].first)
//        ry=-1;
//    else
//        ry=1;
//
//    while (x!=numToMat[pozUrmatoare].second)
//    {
//        x+=rx;
//        if (mat[x][y]==-1)
//            return false;
//    }
```

```
//    while (y!=numToMat[pozUrmatoare].first)
//    {
//        y+=ry;
//        if (mat[x][y]==-1)
//            return false;
//    }
//    return true;


    bool orizontal=false, vertical=false;
    int lastPoz=pozActual;
    for (int i=pozActual+1; i<=pozUrmatoare; ++i)
    {
        if (numToMat[i].second-numToMat[lastPoz].second!=0)
        {
           // if (vertical==true)
             //   return false;
            orizontal=true;
        }
        if (numToMat[i].first-numToMat[lastPoz].first!=0)
        {
            //if (orizontal==false)
              //  return false;
            if (orizontal==true)
                vertical=true;
        }
        lastPoz=i;
    }
    if (orizontal==true && vertical==true)
        return true;
    return false;
}

int urmatoareMultiplu(int dice, int pozActual)
{
//    int aux=dice;
//    while(aux<=52)
//        aux+=dice;
//    aux-=dice;
//    while(aux>=pozActual)
//    {
//        if (mat[numToMat[aux].first][numToMat[aux].second]==0)
//            return aux;
//        aux-=dice;
//    }
//    return -1;
////    int aux=dice;
////    while (aux<=pozActual)
////        aux+=dice;
////    while (mat[numToMat[aux].first][numToMat[aux].second]!=0 &&
aux<53)
////    {
////        aux+=dice;
////    }
```

```
////    if (aux>52)
////        return -1;
////    return aux;
        for (int i=pozActual+dice; i>=pozActual; --i)
        {
            if (i%dice==0 &&
mat[numToMat[i].first][numToMat[i].second]==0)
                return i;
        }
        return -1;
}

void movePlayer(int dice)
{
    sort(pozPlayer+1,pozPlayer+4);
    int pozActual=pozPlayer[1];
    int pozUrmatoare=pozActual+dice;
    ///Verificare daca cade pe o poz ocupata
    if
(mat[numToMat[pozUrmatoare].first][numToMat[pozUrmatoare].second]!=0)
        return;
    if (pozUrmatoare==52)
        {
            pozUrmatoare=88;

//mat[numToMat[pozPlayer[1]].first][numToMat[pozPlayer[1]].second]=0;
            pozActual=0;
            pozPlayer[1]=88;
        }
    else if (ePrim(pozUrmatoare))
    {
        gasireUrmatoarePoz(pozUrmatoare,+1,6, pozActual);
        pozPlayer[1]=pozUrmatoare;
        mat[numToMat[pozActual].first][numToMat[pozActual].second]=0;

mat[numToMat[pozUrmatoare].first][numToMat[pozUrmatoare].second]=1;
    }
    else if (ePatrat(pozUrmatoare))
    {
        gasireUrmatoarePoz(pozUrmatoare,-1,6, pozActual);
        pozPlayer[1]=pozUrmatoare;
        mat[numToMat[pozActual].first][numToMat[pozActual].second]=0;

mat[numToMat[pozUrmatoare].first][numToMat[pozUrmatoare].second]=1;
    }
    else if (rule9(pozUrmatoare, pozActual))
    {
        int v=urmatoareMultiplu(dice, pozActual);
        if (v==-1)
            return;
        else
        {
            pozUrmatoare=v;
            pozPlayer[1]=pozUrmatoare;
```

```
            mat[numToMat[pozActual].first][numToMat[pozActual].second]=0;

mat[numToMat[pozUrmatoare].first][numToMat[pozUrmatoare].second]=1;
        }
    }
    else
    {
        pozPlayer[1]=pozUrmatoare;
       mat[numToMat[pozActual].first][numToMat[pozActual].second]=0;

mat[numToMat[pozUrmatoare].first][numToMat[pozUrmatoare].second]=1;
    }

}

void moveOpponent(int dice)
{
    sort(pozAdversar+1,pozAdversar+4);
    int pozActual=pozAdversar[1];
    int pozUrmatoare=pozActual+dice;
    ///Verificare daca cade pe o poz ocupata
    if
(mat[numToMat[pozUrmatoare].first][numToMat[pozUrmatoare].second]!=0)
        return;
    if (pozUrmatoare==52)
        {
            pozUrmatoare=88;
            pozAdversar[1]=88;
        }
    else if (ePrim(pozUrmatoare))
    {
        gasireUrmatoarePoz(pozUrmatoare,+1,6,pozActual);
        pozAdversar[1]=pozUrmatoare;
        mat[numToMat[pozActual].first][numToMat[pozActual].second]=0;

mat[numToMat[pozUrmatoare].first][numToMat[pozUrmatoare].second]=2;
    }
    else if (ePatrat(pozUrmatoare))
    {
        gasireUrmatoarePoz(pozUrmatoare,-1,6,pozActual);
        pozAdversar[1]=pozUrmatoare;
        mat[numToMat[pozActual].first][numToMat[pozActual].second]=0;

mat[numToMat[pozUrmatoare].first][numToMat[pozUrmatoare].second]=2;
    }
    else if (rule9(pozUrmatoare, pozActual))
    {
        int v=urmatoareMultiplu(dice, pozActual);
        if (v==-1)
            return;
        else
        {
            pozUrmatoare=v;
            pozAdversar[1]=pozUrmatoare;
```

```
            mat[numToMat[pozActual].first][numToMat[pozActual].second]=0;

mat[numToMat[pozUrmatoare].first][numToMat[pozUrmatoare].second]=2;
        }
    }
    else
    {
        pozAdversar[1]=pozUrmatoare;
       mat[numToMat[pozActual].first][numToMat[pozActual].second]=0;

mat[numToMat[pozUrmatoare].first][numToMat[pozUrmatoare].second]=2;
    }
}

int main()
{
    createNumToMat();
    for (int acsl=1; acsl<=5; ++acsl)
    {
        resetMat();
        for (int i=1; i<=3; ++i)
        {
            f >> pozAdversar[i];

mat[numToMat[pozAdversar[i]].first][numToMat[pozAdversar[i]].second]=2;
        }
        for (int i=1; i<=3; ++i)
        {
            f >> pozPlayer[i];

mat[numToMat[pozPlayer[i]].first][numToMat[pozPlayer[i]].second]=1;
        }
        f >> n;
        for (int i=1; i<=n; ++i)
        {
            f >> dice;
            if (i%2==0)
                movePlayer(dice);
            else
                moveOpponent(dice);
//            for (int i=1; i<=3; ++i)
//            {
//                cout << pozPlayer[i] <<' ';
//            }
//            cout << '\n';

        }
        int sPlayer=0, sAdversar=0;
        for (int i=1; i<=52; ++i)
        {
            if (mat[numToMat[i].first][numToMat[i].second]==2)
            {
                sAdversar+=i;
            }
```

```
            else if (mat[numToMat[i].first][numToMat[i].second]==1)
                sPlayer+=i;
        }
        cout <<sAdversar <<' ' << sPlayer << '\n';
//        for (int i=1; i<=3; ++i)
//        {
//            if (pozPlayer[i]!=88)
//                cout << pozPlayer[i] <<' ';
//        }
//        cout << '\n';
    }
    return 0;
}
```