Решение:

Първи начин:

Задачата може да се реши с три едномерни масива – за x, y и цвят c.
Различните отбори се съхраняват в масив otb, като променливата k съхранява колко елемента има попълнени в масива. При всеки нов играч се проверява: има ли го неговият отбор в масива otb и ако е необходимо се добавя. Не трябва да се забравя това действие да се направи и с "любимеца".
За втората част от задачата (намиране на съседа вляво) се обхожда отново масива y и за всеки елемент равен на m, се проверява дали той се среща за пръв път или не (флагът f). Динамично се съхранява най-близкият отляво (пази се за него x-са му в променливата leftx и цвета му в променливата leftc).

```
#include <iostream>

using namespace std;

int main (){
    int N, m, n, i, cl, j;
    cin >> N;
    cin >> m >> n;
    int x[N+1], y[N+1], c[N+1], otb[N+1];
    int k=0, f=0, leftx, leftc;
//въвеждане играчите
    for (i=0; i<N; i++) {
        cin >> x[i] >> y[i] >> c[i];
// проверка има ли такъв отбор в масива otb
        for (j=0; j<k; j++)
            if (otb[j]==c[i]) break;
// записване на отбора в масива otb, ако до момента не е в масива
        if (j==k) otb[k++]=c[i];
    }
    cin >> cl;
// проверка има ли го отбора на любимеца в масива otb
    for (j=0; j<k; j++)
        if (otb[j]==cl) break;
// записване на отбора в масива otb, ако до момента не е в масива
    if (j==k) otb[k++]=c[i];
// извеждане на броя отбори
    cout << k << endl;
// търсене на играч отляво
    for (i=0; i<N; i++)
        if (y[i]==n)
           if (f==0) {
              f=1;
              leftx=x[i];
              leftc=c[i];
           }
           else if (leftx<x[i]) {
                   leftx=x[i];
                   leftc=c[i];
                }
    if (f==0) cout << -1 <<endl;
    else if (leftc==cl) cout << 0 << endl;
         else cout <<leftc <<endl;
    return 0;
}
```

Втори начин:
Данните за всеки от играчите се съхраняват в масива t – представляващ полето за игра. Ако няма играч на дадени координати, то стойността на съответната клетка от t е -2, а ако има играч, то стойността на t е номера на цвета на екипа му. Преброяването на отборите е по-лесно, ако се използва структурата множество, в която поставяме различните цветове.

Намирането на играча отляво може да се получи с обхождане на n-тия ред, от играча в ъгъла - наляво. Ако се намери клетка със стойност различна от -2, то сме открили играч отляво и може да се изведе подходящата стойност – 0, ако е от същият отбор, или цвета на противника. Ако целият ред до началото има нулеви стойности, то никой не заплашва любимеца ни отляво.

```
#include <iostream>
#include <set>
using namespace std;
int main()
{    int N, k, i, m, n, j, col, fl=0;
     //в началото игрището е празно
     int t[640][480];
     for (i=0; i<640; i++)
          for (j=0; j<480;j++)
               t[i][j] = -2;
     //дефиниране на структурата множество
     typedef set<int,greater<int> > IntSet;
     IntSet set1;
     //четене на входните данни
     cin >> N;
     cin >> m >> n;
     for (k = 0; k < N; k++)
      {
      cin >> i;
      cin >> j;
      cin >> t[i][j];
      set1.insert(t[i][j]);
      }
     cin >> t[m][n];
     set1.insert(t[m][n]);
     //извеждане на броят на отборите
     cout <<  set1.size() << endl;
     //намиране на първият отляво
     for (i=m-1; i>=0; i--)
      if (t[i][n]!=-2)
        if (t[i][n] == t[m][n]) {cout << 0; fl=1; break;}
          else {cout << t[i][n]; fl=1;break;}
     if (!fl) cout << - 1;// не е намерил никого вляво
     cout << endl;
     return 0;
}
```