Problem A - Bad Coding

Andrew is a bad coder. As such, he has a weird coding style. We're interested in one of his strangest habits today.
In his code for complex graphs, Andrew inexplicably uses

`pair<pair<int,int>,pair<int,int>>`

as a structure. As such, he can call four different ints for such a structure `p`: `p.first.first`, `p.first.second`, `p.second.first`, and `p.second.second`.

Why does he do this? Why doesn't he just use a tuple like a regular coder? "Because tuples are hard. Typing `get<0>(t)` is so much work," he says.
However, this created problems for Andrew. One day, he accidentally deleted his initialization of `p`, and was left staring at all the ints he had called within `p`. Thankfully, he had called every single `int` exactly once. Can you help determine what the structure of `p` is?

```
void search(pii p, int i, int t){
    memset(dijk[i][t],-1,sizeof dijk[i][t]);
    dijk[i][t][p.first][p.second][t]=0;
    set<pair<int,pair<pii,int>>> pq;
    pq.insert({0,{p,t}});
    while (!pq.empty()){
        pair<int,pair<pii,int>> asd=*pq.begin(); pq.erase(pq.begin());
        pair<pii,int> now=asd.second;
        for (int j=0;j<4;j++){
            pii dd=delta[(org[now.first.first][now.first.second]+now.second+j)%4];
            pair<pii,int> nex;
            nex.first={now.first.first+dd.first,now.first.second+dd.second};
            if (!inRange(nex.first)) continue;
            int newd=dijk[i][t][now.first.first][now.first.second][now.second]+j+1;
            nex.second=(now.second+j+1)%4;
            if ((newd<dijk[i][t][nex.first.first][nex.first.second][nex.second])||
            (dijk[i][t][nex.first.first][nex.first.second][nex.second]==-1)){
                pq.erase({dijk[i][t][nex.first.first][nex.first.second][nex.second],nex});
                dijk[i][t][nex.first.first][nex.first.second][nex.second]=newd;
                pq.insert({dijk[i][t][nex.first.first][nex.first.second][nex.second],nex});
            }
        }
    }
}
```

16:19 C and C++

Input Specification:

The input will start with a single integer T, the number of test cases.
Each test case will start with one line $1 \leq N \leq 10^5$, denoting how many `int` variables are within `p`. Then N lines follow listing the different variables `p` contains, where `.f` means `.first` and `.s` means `.second`. It is guaranteed that this input set is consistent with some structure of `p`.

Output Specification:

For each test case, please output the structure of `p` using the format below. The output of each test case should be on a separate line.

Sample Input:

```
1
4
p.f
p.s.f.f
p.s.f.s
p.s.s
```

Sample Output:

```
pair<int,pair<pair<int,int>,int>>
```