```
/*** 10051 - Tower of Cubes:
You are given N colorful cubes each having a distinct weight. Each face of a
cube is colored with one color. Your job is to build a tower using the cubes
you have subject to the following restrictions:
 1) Never put a heavier cube on a lighter one.
 2) The bottom face of every cube(except the bottom cube,which is lying on
   the floor) must have the same color as the top face of the cube below it.
 3) Construct the tallest tower possible.

The first line of each test case contains an integer N(1<=N<=500) indicating
the number of cubes you are given. The i’th (1<=i<=N) of the next N lines
contains the description of the i’th cube. A cube is described by giving the
colors of its faces in the following order: front, back, left, right, top and bottom face.
For your convenience colors are identified by integers in the range 1 to 100.
You may assume that cubes are given in the increasing order of their weights,
that is, cube 1 is the lightest and cube N is the heaviest.
The input terminates with a value 0 for N.

Print the number of cubes in the tallest tower you have built. From the next line
describe the cubes in your tower from top to bottom with one description per line.
Each description contains an integer (giving the serial number of this cube in
the input) followed by a single whitespace character and then the identification
string (front, back, left, right, top or bottom) of the top face of the cube in
the tower. Note that there may be multiple solutions and any one of them is acceptable.
****/
int n,k,a[505][10],dp[6025]; vector<int>ed[6025],vv;
struct dt{ int id,tp,bo,lf,rg,fr,ba; }st[6025];
void nodeCreate(){
    k=0;
    for(int i=n; i>=1; i--){
        st[++k].id=i;
        st[k].tp=1; st[k].bo=2; st[k].lf=3;
        st[k].rg=4; st[k].fr=6; st[k].ba=5;

        st[++k].id=i;
        st[k].tp=1; st[k].bo=2; st[k].lf=4;
        st[k].rg=3; st[k].fr=5; st[k].ba=6;

        st[++k].id=i;
        st[k].tp=2; st[k].bo=1; st[k].lf=4;
        st[k].rg=3; st[k].fr=6; st[k].ba=5;

        st[++k].id=i;
        st[k].tp=2; st[k].bo=1; st[k].lf=3;
        st[k].rg=4; st[k].fr=5; st[k].ba=6;

        st[++k].id=i;
        st[k].tp=3; st[k].bo=4; st[k].lf=6;
        st[k].rg=5; st[k].fr=1; st[k].ba=2;

        st[++k].id=i;
        st[k].tp=3; st[k].bo=4; st[k].lf=5;
        st[k].rg=6; st[k].fr=2; st[k].ba=1;

        st[++k].id=i;
        st[k].tp=4; st[k].bo=3; st[k].lf=5;
        st[k].rg=6; st[k].fr=1; st[k].ba=2;

        st[++k].id=i;
        st[k].tp=4; st[k].bo=3; st[k].lf=6;
        st[k].rg=5; st[k].fr=2; st[k].ba=1;

        st[++k].id=i;
        st[k].tp=5; st[k].bo=6; st[k].lf=3;
        st[k].rg=4; st[k].fr=1; st[k].ba=2

        st[++k].id=i;
        st[k].tp=5; st[k].bo=6; st[k].lf=4;
        st[k].rg=3; st[k].fr=2; st[k].ba=1;

        st[++k].id=i;
        st[k].tp=6; st[k].bo=5; st[k].lf=4;
        st[k].rg=3; st[k].fr=1; st[k].ba=2;

        st[++k].id=i;
        st[k].tp=6; st[k].bo=5; st[k].lf=3;
        st[k].rg=4; st[k].fr=2; st[k].ba=1;
    }
}
void graphCreate(){
    for(int i=1; i<=k; i++){
        for(int j=i+1; j<=k; j++){
            if(st[i].id==st[j].id)continue;
            if(a[st[i].id][st[i].tp]==a[st[j].id][st[j].bo]){
                ed[i].push_back(j);
            }
        }
    }
    for(int i=1; i<=k;i++) ed[0].push_back(i);
}
int lis(int u){
    if(ed[u].size()==0)return dp[u]=1;
    if(dp[u]!=-1)return dp[u];
    int ret = 0;
    for(int i=0; i<ed[u].size(); i++){
        int v = ed[u][i];
        ret = max(ret,1+lis(v));
    }
    return dp[u] = ret;
}
void path(int u,int x){
    if(x==0)return;
    for(int i=0; i<ed[u].size(); i++){
        int v = ed[u][i];
        int ret = lis(v);
        if(ret==x){
            vv.push_back(v);
            path(v,x-1);
            break;
        }
    }
}
int main(){
    int ks=0;
    while(scanf("%d",&n) && n){
        for(int i=1; i<=n; i++){
            for(int j=1; j<=6; j++){
                scanf("%d",&a[i][j]);
            }
        }

        if(ks>0)printf("\n");
        printf("Case #%d\n",++ks);

        nodeCreate();
        graphCreate();

        memset(dp,-1,sizeof(dp));
        int ans = lis(0);
        printf("%d\n",ans-1);

        path(0,ans-1);

        for(int i=vv.size()-1; i>=0; i--){
            int nd = vv[i]; int id = st[nd].id; int tp = st[nd].tp;
            printf("%d ",id);
            if(tp==1)printf("front\n");
            else if(tp==2)printf("back\n");
            else if(tp==3)printf("left\n");
            else if(tp==4)printf("right\n");
            else if(tp==5)printf("top\n");
            else if(tp==6)printf("bottom\n");
        }

        vv.clear();
        for(int i=0; i<=k; i++)ed[i].clear();
    }
}
```