/\*

本题实际获胜情况并不多，因此可以直接使用状态压缩搜索+优化来解决

优化部分：状态压缩判断棋盘有几条连线

使用广搜+滚动数组+hash优化内存

\*/

#include<bits/stdc++.h>

#define P pair<int,int>

using namespace std;

const int mod=1e9+7;

int m=9998479;

int harry\_board[26];

int hermione\_board[26];

int hermione\_order[26];

long long node\_bit[26];

long long one=1;

inline int low\_bit(int x){return x&-x;}

void print\_bit(long long a)

{

while(a)

{

cout<<a%2;

a/=2;

}

cout<<endl;

}

void bit\_set()

{

int line\_number=0;

for(int x=1;x<=5;x++)

{

line\_number++;

for(int y=0;y<=4;y++)

{

node\_bit[y\*5+x]|=(one<<(((line\_number-1)\*5)+y));

}

}

for(int y=0;y<=4;y++)

{

line\_number++;

for(int x=1;x<=5;x++)

{

node\_bit[y\*5+x]|=(one<<(((line\_number-1)\*5)+x-1));

}

}

line\_number++;

for(int i=1;i<=5;i++)node\_bit[(i-1)\*5+i]|=(one<<(((line\_number-1)\*5)+i-1));

line\_number++;

for(int i=1;i<=5;i++)node\_bit[(i-1)\*5+5-i+1]|=(one<<(((line\_number-1)\*5)+i-1));

return;

}

int count\_line(long long state)

{

int cnt=0;

while(state)

{

if((state&31)==31)cnt++;

state>>=5;

}

return cnt;

}

int judge(int node\_state)

{

long long harry\_state=0;

long long hermione\_state=0;

int cnt=0;

while(node\_state)

{

cnt++;

if(node\_state%2==1)

{

harry\_state|=node\_bit[harry\_board[hermione\_order[cnt]]];

hermione\_state|=node\_bit[hermione\_board[hermione\_order[cnt]]];

}

node\_state>>=1;

}

int harry=count\_line(harry\_state);

int hermione=count\_line(hermione\_state);

if(harry>=5&&hermione<5)return 1;

if(hermione>=5)return -1;

return 0;

}

void bit\_test()

{

int test[26]={0,1,6,11,16,21,7,13,19,25,4,9,14,24};

long long test\_l=0;

for(int i=1;i<=25;i++)test\_l|=node\_bit[test[i]];

int cnt=1;

cout<<" sum of line : "<<count\_line(test\_l)<<endl;

while(test\_l!=0)

{

cout<<(test\_l&1);

test\_l>>=1;

cout<<" - "<<cnt<<endl;

cnt++;

}

}

int ans=0;

int f[10000000][2]={0};

void BFS()

{

queue<P> q;

q.push(P(0,0));

f[0][0]=1;

int step=0;

while(!q.empty())

{

P p=q.front();

q.pop();

int state=p.first;

int j\_line=judge(state);

int k=p.second;

if(j\_line==-1)

{

f[state%m][k]=0;

continue;

}

if(j\_line==1)

{

ans+=f[state%m][k];

ans%=mod;

f[state%m][k]=0;

continue;

}

if(k%2==0)

{

for(int i=0;i<25;i++)

{

int node=1<<i;

if(state&node)continue;

if(f[(state|node)%m][k^1]==0)q.push(P(state|node,k^1));

f[(state|node)%m][k^1]=(f[state%m][k]+f[(state|node)%m][k^1])%mod;

}

}

if(k%2==1)

{

for(int i=0;i<25;i++)

{

int node=1<<i;

if((state&node)==0)

{

if(f[(state|node)%m][k^1]==0)q.push(P(state|node,k^1));

f[(state|node)%m][k^1]=(f[state%m][k]+f[(state|node)%m][k^1])%mod;

break;

}

}

}

f[state%m][k]=0;

}

}

void test()

{

long long state=0;

for(int i=1;i<=25;i++)

{

state+=node\_bit[i];

}

print\_bit(state);

}

int main()

{

// freopen("bingo.in","r",stdin);

int T;

bit\_set();

cin>>T;

while(T--)

{

int a;

for(int i=1;i<=25;i++){scanf("%d",&a),harry\_board[a]=i;}

for(int i=1;i<=25;i++){scanf("%d",&a),hermione\_board[a]=i;}

for(int i=1;i<=25;i++)scanf("%d",&hermione\_order[i]);

BFS();

cout<<ans<<endl;

ans=0;

}

}