```
#include<bits/stdc++.h>
#define int long long
#define re register
#define inf 0x7fffffff
const int L=1<<20|1;
char buffer[L],*S,*T;
#define getchar() ((S==T&&(T=(S=buffer)+fread(buffer,1,L,stdin),S==T))?EOF:*S++)
using namespace std;
int n,m,tot,cmpid,root,X,Y;
inline int read(){
        re int a=0,b=1; re char ch=getchar();
        while(ch<'0'||ch>'9')
                b=(ch=='-')?-1:1,ch=getchar();
        while(ch>='0'&&ch<='9')
                a=(a<<3)+(a<<1)+(ch^48),ch=getchar();
        return a*b;
}
struct node{int dis,id;};
inline bool operator < (node a,node b){
        return a.dis>b.dis||(a.dis==b.dis&&a.id<b.id);
}
priority_queue<node> q;
struct point{
        int x[2],id;
        friend bool operator < (const point &a,const point &b)
                {return a.x[cmpid]<b.x[cmpid];}
}p[100010];
struct tree{
        point p;int mx[2],mn[2],ls,rs,id;
}t[100010];
inline int dis(tree x){
        re int P=(x.p.x[0]-X)*(x.p.x[0]-X);
        re int Q=(x.p.x[1]-Y)*(x.p.x[1]-Y);
        return P+Q;
}
inline int mxdis(tree x){
        re int P=(x.mn[0]-X)*(x.mn[0]-X);
        re int M=(x.mx[0]-X)*(x.mx[0]-X);
        re int Q=(x.mn[1]-Y)*(x.mn[1]-Y);
        re int N=(x.mx[1]-Y)*(x.mx[1]-Y);
        return max(P,M)+max(Q,N);
}
inline void update(re int x){
        if(!x)  return ; re int l=t[x].ls,r=t[x].rs;
        if(l) t[x].mn[0]=min(t[x].mn[0],t[l].mn[0]),
                  t[x].mn[1]=min(t[x].mn[1],t[l].mn[1]),
                  t[x].mx[0]=max(t[x].mx[0],t[l].mx[0]),
                  t[x].mx[1]=max(t[x].mx[1],t[l].mx[1]);
        if(r) t[x].mn[0]=min(t[x].mn[0],t[r].mn[0]),
                  t[x].mn[1]=min(t[x].mn[1],t[r].mn[1]),
                  t[x].mx[0]=max(t[x].mx[0],t[r].mx[0]),
                  t[x].mx[1]=max(t[x].mx[1],t[r].mx[1]);
}
inline void query(re int x){
        if(!x) return ;
        re int res=dis(t[x]);
        if(res>q.top().dis||(res==q.top().dis&&t[x].id<q.top().id))
                q.pop(),q.push((node){res,t[x].id});
        re int l=t[x].ls,r=t[x].rs,ld,rd;
        if(l) ld=mxdis(t[l]);
        if(r) rd=mxdis(t[r]);
        if(ld>rd){
```

```
                if(ld>=q.top().dis) query(l);
                if(rd>=q.top().dis) query(r);
        }
        else{
                if(rd>=q.top().dis) query(r);
                if(ld>=q.top().dis) query(l);
        }
}
inline void build(re int &x,re int l,re int r,re int k){
        if(l>r) return ;
        x=++tot;cmpid=k;
        re int mid=(l+r)>>1;
        nth_element(p+l,p+mid,p+r+1);
        t[x].p=p[mid];t[x].id=t[x].p.id;
        t[x].mn[0]=t[x].mx[0]=t[x].p.x[0];
        t[x].mn[1]=t[x].mx[1]=t[x].p.x[1];
        build(t[x].ls,l,mid-1,k^1);
        build(t[x].rs,mid+1,r,k^1);
        update(x);
}
signed main(){
        n=read();
        for(re int i=1;i<=n;i++)
                p[i].x[0]=read(),p[i].x[1]=read(),p[i].id=i;
        build(root,1,n,0);
        m=read();
        for(re int i=1,k;i<=m;i++){
                X=read(),Y=read(),k=read();
                while(q.size()) q.pop();
                for(re int j=1;j<=k;j++) q.push((node){-1,0});
                query(root);
                printf("%lld\n",q.top().id);
        }
        return 0;
}
```