Convex Hull-Basic Code

```
using namespace std;
class point{
public:
   ll x,y;
};
point bindu[mx],Pivot;
stack<point>st;
ll orientation(point a,point b,point c){
   return ((b.y-a.y)*(c.x-b.x))-((c.y-b.y)*(b.x-a.x));
}
ll dis(point a,point b){
   return ((b.x-a.x)*(b.x-a.x))+((b.y-a.y)*(b.y-a.y));
}
bool cmp(point a,point b){
   if(orientation(Pivot,a,b)==0){//checking co-linearity
      return dis(Pivot,a)<dis(Pivot,b);//if co-linear put nearest one
   }
   ll m1x=a.x-Pivot.x,m1y=a.y-Pivot.y;
   ll m2x=b.x-Pivot.x,m2y=b.y-Pivot.y;
   return (atan2((double)m1y,(double)m1x)-atan2((double)m2y,(double)m2x)<0);
}
point nextToTop(){
   point a=st.top();
   st.pop();
   point b=st.top();
   st.push(a);
   return b;
}
void convexHull(ll n){
   ll ymin=bindu[0].y,mn=0;
   for(int i=1;i<n;i++){//Finding the bottom-most point
      if(bindu[i].y<ymin||(ymin==bindu[i].y&&bindu[i].x<bindu[mn].x)){
         ymin=bindu[i].y,mn=i;
      }
   }
   swap(bindu[0],bindu[mn]);//place the bottom-most point in the 1st position
   Pivot=bindu[0];
   sort(bindu,bindu+n,cmp);
   ll m=1;
   for(int i=1;i<n;i++){
      while(i<n-1&&orientation(Pivot,bindu[i],bindu[i+1])==0) i++;//removing same angle points
      bindu[m++]=bindu[i];
   }
   cout<<m-1<<endl;
   if(m<3) return;//Convex Hull is not possible
   st.push(bindu[0]);
   st.push(bindu[1]);
```

```
    st.push(bindu[2]);
    for(int i=3;i<m;i++){
        while(orientation(nextToTop(),st.top(),bindu[i])>=0) st.pop();
        st.push(bindu[i]);
    }
    while(!st.empty()){
        point a=st.top();
        cout<<a.x<<" "<<a.y<<endl;
        st.pop();
    }
}
int main(){
//    freopen("Input.txt","r",stdin); freopen("Output.txt","w",stdout);
//    ios_base::sync_with_stdio(false); cin.tie(NULL);
    ll n;
    scanf("%lld",&n);
    for(int i=0;i<n;i++){
        scanf("%lld%lld",&bindu[i].x,&bindu[i].y);
    }
    convexHull(n);
    return 0;
}
/**
Input:
8
0 3 1 1 2 2 4 4 0 0 1 2 3 1 3 3
Output:
4
0 3
4 4
3 1
0 0
**/
```