E. Domino Principle

time limit per test 2 seconds
memory limit per test 256 megabytes
input standard input
output standard output

Vasya is interested in arranging dominoes. He is fed up with common dominoes and he uses the dominoes of different heights. He put n dominoes on the table along one axis, going from left to right. Every domino stands perpendicular to that axis so that the axis passes through the center of its base. The i-th domino has the coordinate x_i and the height h_i. Now Vasya wants to learn for every domino, how many dominoes will fall if he pushes it to the right. Help him do that.

Consider that a domino falls if it is touched strictly above the base. In other words, the fall of the domino with the initial coordinate x and height h leads to the fall of all dominoes on the segment $[x + 1, x + h - 1]$.

Input

The first line contains integer n ($1 \le n \le 10^5$) which is the number of dominoes. Then follow $n$lines containing two integers x_i and h_i ($-10^8 \le x_i \le 10^8, 2 \le h_i \le 10^8$) each, which are the coordinate and height of every domino. No two dominoes stand on one point.

Output

Print n space-separated numbers z_i — the number of dominoes that will fall if Vasya pushes the i-th domino to the right (including the domino itself).

Examples

input Copy

```
4
16 5
20 5
10 10
18 2
```

output Copy

```
3 1 4 1
```

input Copy

```
4
0 10
1 5
9 10
15 10
```

output Copy

```
4 1 2 1
```

```cpp
/// http://codeforces.com/contest/56/problem/E
#include<bits/stdc++.h>
using namespace std;
#define mx 200005
int tree[4*mx], cs[mx], ar[mx];
struct dt{ int b,e,h,id,res; }st[100005];
bool cmp1(dt x,dt y){ return x.b<y.b; }
bool cmp2(dt x,dt y){ return x.id<y.id; }

void update(int nd,int b,int e,int x,int v){
    if(b>x || e<x)return;
    if(b==x && e==x){ tree[nd]=v; return; }
    int left = 2*nd; int right= 2*nd+1; int md = (b+e)/2;
    update(left,b,md,x,v);
    update(right,md+1,e,x,v);
    tree[nd] = max(tree[left],tree[right]);
}

int query(int nd,int b,int e,int x,int y){
    if(e<x || b>y)return 0;
    if(b>=x && e<=y) return tree[nd];
    int left = 2*nd; int right= 2*nd+1; int md = (b+e)/2;
    int p1 = query(left,b,md,x,y);
    int p2 = query(right,md+1,e,x,y);
    return max(p1,p2);
}

int main(){
    int n; scanf("%d",&n);
    set<int>ss;
    for(int i=1; i<=n; i++){
        int b,h,e; scanf("%d%d",&b,&h);
        e = b+h-1; st[i].b=b; st[i].e=e;
        st[i].h=h; st[i].id=i; st[i].res=-1;
        ss.insert(b); ss.insert(e);
    }

    map<int,int>mp;
    set<int> :: iterator it;
    int m=0;
    for(it=ss.begin(); it!=ss.end(); it++) mp[*it]=++m;

    sort(st+1,st+n+1,cmp1);
    int sz = ss.size();
    for(int i=1; i<=n; i++){
        int v = mp[st[i].b];
        ar[v]=1;
        int u = mp[st[i].e];
        update(1,1,sz,v,u);
    }

    cs[0]=0;
    for(int i=1; i<=sz; i++) cs[i]=cs[i-1]+ar[i];

    for(int i=n; i>=1; i--){
        int b = mp[st[i].b];
        int e = mp[st[i].e];
        int p = query(1,1,sz,b,e);
        int res = cs[p]-cs[b-1];
        st[i].res = res;
        update(1,1,sz,b,p);
    }

    sort(st+1,st+n+1,cmp2);
    for(int i=1; i<=n; i++){
        if(i==n) printf("%d\n",st[i].res);
        else printf("%d ",st[i].res);
    }
    return 0;
}
```