```cpp
/*
    | Segment Tree - Lazy Propagation |
    Desc: Range Update Range Queries in O(n*log(n)).
    Source: KawakiMeido
    State: Untested lmao
*/

struct LazySegmentTree{
    struct Node{
        int val;
        Node(){
            val = INF;
        }
    };

    int n;
    vector<Node> IT;
    vector<int> lazy;

    void add(int idx, int x){

        //Update logic

    }

    void propagate(int idx){
        add(idx*2,lazy[idx]);
        add(idx*2+1,lazy[idx]);
        lazy[idx] = 0;
    }

    Node comb(Node l, Node r){
        if (l.val == INF) return r;
        if (r.val == INF) return l;

        Node i;

        //Update logic

        return i;
    }

    void build(int idx, int l, int r){
        if (l==r){

            //Update logic

            return;
        }
```

```
50
51          int mid = (l+r)/2;
52          build(idx*2,l,mid);
53          build(idx*2+1,mid+1,r);
54          IT[idx] = comb(IT[idx*2],IT[idx*2+1]);
55      }
56
57      void update(int idx, int l, int r, int x, int y, int val){
58          if (y < l || r < x) return;
59          if (x <= l && r <= y){
60              add(idx,val);
61              return;
62          }
63
64          propagate(idx);
65
66          int mid = (l+r)/2;
67          update(idx*2,l,mid,x,y,val);
68          update(idx*2+1,mid+1,r,x,y,val);
69          IT[idx] = comb(IT[idx*2],IT[idx*2+1]);
70      }
71
72      Node getNode(int idx, int l, int r, int x, int y){
73          if (y < l || r < x) return Node();
74          if (x <= l && r <= y){
75              return IT[idx];
76          }
77
78          propagate(idx);
79
80          int mid = (l+r)/2;
81          return comb(getNode(idx*2,l,mid,x,y),getNode(idx*2+1,mid+1,r,x,y));
82
83      }
84
85      void init(int _n){
86          n = _n;
87          IT.resize(n*4+10, Node());
88          lazy.resize(n*4+10, 0);
89          build(1,1,n);
90      }
91  };
92
```