Dynamic segment tree with lazy:

```cpp
#define left l , (l+r)>>1
#define right ((l+r)>>1)+1 , r

struct DynamicSegTree{
    struct Vertex {
        int value , lazy , l , r;
        Vertex *lChild, *rChild;

        Vertex(int value , int l , int r) :
            lChild(nullptr), rChild(nullptr),
            value(value), lazy(0) , l(l) , r(r){}

        void extend() {
            if(lChild==nullptr){
                lChild = new Vertex(0 , left);
                rChild = new Vertex(0 , right);
            }
        }
    };
    Vertex *root;
    int n;

    DynamicSegTree(int n) : n(n){
        root = new Vertex(0 , 0 , n);
    }

    void push(Vertex *p){
        if(!(p->lazy)) return;
        //product in size of segment if you want all element to inc
        p->lChild->value+=p->lazy; p->rChild->value+=p->lazy;
        p->lChild->lazy+=p->lazy;  p->rChild->lazy+=p->lazy;
        p->lazy = 0;
    }

    void update(int i , int j , int val){
        update(i , j , val , root);
    }

    int update(int i , int j , int val , Vertex *p){
        if(j<p->l || p->r<i) return p->value;
        if(i<=p->l && p->r<=j){
            p->lazy+=val; //to change
            p->value+=val; //to change
            return p->value;
        }
        p->extend(); push(p);
        return p->value = (update(i , j , val , p->lChild) +
                           update(i , j , val , p->rChild)); //to change
    }

    int query(int i , int j){
        return query(i , j , root);
    }

    int query(int i , int j , Vertex *p){
        if(j<p->l || p->r<i) return 0; //to change
        if(i<=p->l && p->r<=j) return p->value;
        p->extend(); push(p);
        return (query(i , j , p->lChild) + query(i , j , p->rChild)); //to change
    }
};
```