Persistent Segment Tree:

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

struct PersistentSegTree{
    struct Vertex {
        Vertex *l, *r;
        int sum;

        Vertex(int val) : l(nullptr), r(nullptr), sum(val) {}
        Vertex(Vertex *l, Vertex *r) : l(l), r(r), sum(0) {
            if (l) sum += l->sum;
            if (r) sum += r->sum;
        }
        //to change marge
    };

    int n;
    vector<int> &a;
    vector<Vertex*> roots;

    PersistentSegTree(int n  , vector<int> &a) : a(a) {
        this->n = n;
        roots.push_back(build(0 , n-1));
    }

    Vertex* build(int l , int r){
        if(l==r) return new Vertex(a[l]);
        return new Vertex(build(left) , build(right));
    }

    //point update
    void update(int i , int val , int rootIndex){
        roots.push_back(
            update(i , val , roots[rootIndex] , 0 , n-1)
        );
    }

    Vertex* update(int i , int val , Vertex* p , int l , int r){
        if(l==r) return new Vertex(val); //to change
        if(i <= (l+r)>>1)
            return new Vertex(update(i , val , p->l , left) , p->r);
        else
            return new Vertex(p->l , update(i , val , p->r , right));
    }

    //use this function
    int query(int i , int j , int rootIndex){
        return query(i , j , roots[rootIndex] , 0 , n-1);
    }

    int query (int i , int j , Vertex* p , int l , int r){
        if(j<l  ||  r<i) return 0; // to change
        if(i<=l && r<=j) return p->sum;
        return (query(i , j , p->l , left) + query(i , j , p->r , right)); //to change
    }
};
```