```cpp
#include <bits/stdc++.h>
#define ll long long
#define inf (int)1e9
using namespace std;
```

Segment Tree functions:

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

vector<int> seg(400100) , a(100100) , lazy(400100);

int build(int p , int l , int r){
    if(l==r) return seg[p] = a[l];
    return seg[p] = (build(left) + build(right)); //to change
}

void push(int p){
    if(!lazy[p]) return;
    seg[p<<1] += lazy[p]; seg[p<<1|1] += lazy[p];
    lazy[p<<1] += lazy[p]; lazy[p<<1|1] +=lazy[p];
    lazy[p] = 0;
}

int update(int i , int j , int inc , int p , int l , int r){
    if(j<l || r<i) return seg[p];
    if(i<=l && r<=j) return lazy[p] += inc, seg[p] += inc; //to change
    push(p);
    return seg[p] = (update(i , j , inc , left) + update(i , j , inc , right)); //to change
}

int query (int i , int j , int p , int l , int r){
    if(j<l  ||  r<i) return inf; // to change
    if(i<=l && r<=j) return seg[p];
    push(p);
    return (query(i , j , left) + query(i , j , right)); //to change
}

//binary search inside segment tree
int query(int k , int p , int l , int r){
    // check not found case
    if(seg[p] < k) return -1;

    // return the answer
    if(l==r) return l;

    // check the condition and go left or right
    if(seg[p<<1] >= k){
        return query(k , left);
    }
    return query(k-seg[p<<1] , right);
}
```