```cpp
#include<bits/stdc++.h>
using namespace std;
//array declared globally so that we don't have to pass them again and again
long input[4]={3,2,4,1}; //taking an example array input, we will build a minimum range query segment tree
long seg[16]; //Segment tree array, its size in worst case might get 4*n
//function prototypes
void printarray(long [],long); //A function to print array
long query(long ,long ,long ,long ,long);
void build(long tnode,long starti,long endi) //building the segment tree
{
    if(starti==endi) //if index is like (5,5), the value to be fetched is input[5], by common sense
    {
        seg[tnode]=input[starti];
        return;
    }
    else //else build the childs and then get value
    {
        long mid=(starti+endi)/2; //calculate mid
        build(tnode*2,starti,mid); //build left subtree
        build(tnode*2+1,mid+1,endi); //build right subtree
        if(seg[tnode*2]<seg[tnode*2+1]) //fetch values from children
        {
            seg[tnode]=seg[tnode*2];
        }
        else
        {
            seg[tnode]=seg[tnode*2+1];
        }
    }
}
int main()
{
    build(1,0,3); //1 is start index of segtree, 0 and 3 is the range of input array
    printarray(seg,16); // printing the segtree array
    cout<<query(1,0,3,0,2); //1 start index of segtree, 0 and 3 range of input array,0 and 2 range of query
    return 0;
}
void printarray(long input[],long n)
{
```

```
    long i;

    cout<<"-----------------------------------------------------"<
    <endl;
    for(i=0;i<n;i++)
    {
        cout<<input[i]<<" ";
    }

    cout<<endl<<"-----------------------------------------------
    ----"<<endl;
}
long query(long node,long l,long r,long qs,long qe)
{
    if(qs<=l && qe>=r) //if falls under query range
    {
        return seg[node];
    }
    else if(qe<l || qs>r) //if falls out of query range
    {
        return INT_MAX;
    }
    else //else
    {
        long mid=(l+r)/2; //take mid
        long a=query(node*2,l,mid,qs,qe); //build two halves
        of the tree
        long b=query(node*2+1,mid+1,r,qs,qe);
        if(a<b) //return the smaller value
        {
            return a;
        }
        else
        {
            return b;
        }
    }
}
```