#include<bits/stdc++.h>

**using** **namespace** std**;**

**typedef** vector**<**int**>** vi**;**

class SegmentTree **{** // the segment tree is stored like a heap array

public**:** vi st**,** A**;** // recall that vi is: typedef vector<int> vi;

// Le A: le vecteur de base

// le st : segment tre arbre sous forme de vector: (!) seul le st est 1-based le reste est 0 based

// We save Indexes !!!!!

int n**;**

int qwery**(**int p1**,** int p2**)**//ce sont des indexes !!!!

**{**

**return** **(**A**[**p1**]** **<=** A**[**p2**])** **?** p1 **:** p2**;**

**}**

int left **(**int p**)** **{** **return** p **<<** 1**;** **}** // same as binary heap operations

int right**(**int p**)** **{** **return** **(**p **<<** 1**)** **+** 1**;** **}**

void build**(**int p**,** int L**,** int R**)** **{** // O(n)

**if** **(**L **==** R**)** // as L == R, either one is fine

st**[**p**]** **=** L**;** // store the index

**else** **{** // recursively compute the values

build**(**left**(**p**)** **,** L **,** **(**L **+** R**)** **/** 2**);**

build**(**right**(**p**),** **(**L **+** R**)** **/** 2 **+** 1**,** R **);**

int p1 **=** st**[**left**(**p**)],** p2 **=** st**[**right**(**p**)];**

st**[**p**]** **=** qwery**(**p1**,** p2**);**

**}** **}**

int rmq**(**int p**,** int L**,** int R**,** int i**,** int j**)** **{** // O(log n)

**if** **(**i **>** R **||** j **<** L**)** **return** **-**1**;** // current segment outside query range

**if** **(**L **>=** i **&&** R **<=** j**)** **return** st**[**p**];** // inside query range

// compute the min position in the left and right part of the interval

int p1 **=** rmq**(**left**(**p**)** **,** L **,** **(**L**+**R**)** **/** 2**,** i**,** j**);**

int p2 **=** rmq**(**right**(**p**),** **(**L**+**R**)** **/** 2 **+** 1**,** R **,** i**,** j**);**

**if** **(**p1 **==** **-**1**)** **return** p2**;** // if we try to access segment outside query

**if** **(**p2 **==** **-**1**)** **return** p1**;** // same as above

**return** qwery**(**p1**,** p2**);** // as in build routine

**}**

public**:**

SegmentTree**(**const vi **&**\_A**)** **{**

A **=** \_A**;** n **=** **(**int**)**A**.**size**();** // copy content for local usage

st**.**assign**(**4 **\*** n**,** 0**);** // create large enough vector of zeroes

build**(**1**,** 0**,** n **-** 1**);** // recursive build

**}**

int rmq**(**int i**,** int j**)** **{** **return** rmq**(**1**,** 0**,** n **-** 1**,** i**,** j**);** **}** // overloading

private**:**

int update**(**int pos**,** int p**,** int value**,** int L**,** int R**){** // O(log n)

**if** **(**pos **>** R **||** pos **<** L**)** **return** st**[**p**];** // current segment outside query range

**if** **(**L **==** pos **&&** R **==** pos**)** **{**

cerr**<<**A**[**pos**]<<**"\_\_"**;**

A**[**pos**]=**value**;**

cerr**<<**A**[**pos**]<<**"\_\_"**;**

**return** st**[**p**];}** // INDEXXX fel st!!! inside query range

// compute the min position in the left and right part of the interval

int p1 **=** update**(**pos**,** left**(**p**)** **,**value**,** L **,** **(**L**+**R**)** **/** 2**);**

int p2 **=** update**(**pos**,** right**(**p**),**value**,** **(**L**+**R**)** **/** 2 **+** 1**,** R**);**

// meme les segments outside query sont utilisé pour màj la branche

**return** st**[**p**]=**qwery**(**p1**,** p2**);** // as in build routine

**}**

public**:**

void update**(**int pos**,** int value**){**

//p=position dans st, pos=position dans A

update**(**pos**,** 1**,** value**,** 0**,** n**-**1**);**

**}**

**};**

int main**()** **{**

int arr**[]** **=** **{** 18**,** 17**,** 13**,** 19**,** 15**,** 11**,** 20 **};** // the original array

vi A**(**arr**,** arr **+** 7**);**

SegmentTree st**(**A**);**

printf**(**"RMQ(1, 3) = %d\n"**,** st**.**rmq**(**1**,** 3**));** // answer = index 2, zero based

printf**(**"RMQ(4, 6) = %d\n"**,** st**.**rmq**(**4**,** 6**));** // answer = index 5

**for(**auto**&** e **:** st**.**A**)** cerr**<<**e**<<**" "**;**

cerr**<<**endl**;**

**for(**auto**&** e **:** st**.**st**)** cerr**<<**e**<<**" "**;**

st**.**update**(**1**,**5**);**// 0 - based

cerr**<<**endl**;**

**for(**auto**&** e **:** st**.**st**)** cerr**<<**e**<<**" "**;**

cerr**<<**endl**;**

**for(**auto**&** e **:** st**.**A**)** cerr**<<**e**<<**" "**;**

printf**(**"RMQ(1, 3) = %d\n"**,** st**.**rmq**(**1**,** 3**));** // answer = index 1

**}** // return 0;