**CLIMBING LEADERBOARD**

#include<assert.h>

#include<limits.h>

#include<math.h>

#include<stdbool.h>

#include<stddef.h>

#include<stdint.h>

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int main (void)

{

    int i = 0 ;

    int n = 0 ;

    scanf("%d",&n);

    int stack[2000002];

    int score = 0 ;

    int top = 0 ;

    for(i = 0; i < n; ++i)

    {

        scanf("%d",&stack[top+1]) ;

        if(stack[top+1]!=stack[top])

        {

            ++top ;

        }

    }

    int m = 0 ;

    scanf("%d",&m);

    for( i = 0; i < m; ++i)

    {

        scanf("%d",&score);

        while(top && score >=stack[top])

        {

            --top ;

        }

            printf("%d\n",top+1) ;

    }

return 0 ;

}

#include <stdio.h>

// variables needed throughout the code

int i, n, m, score;

// the stack is implemented using an array

// and a variable storing the index of the top element

// note: global variables are automatically intialized to 0,

// so top is 0 and the elements of the stack are 0

int top, stack[200002];

int main() {

// read n - the number of players on the leaderboard

scanf("%d", &n);

// read players' scores

// and store the ones that are distinct

for (i = 0; i < n; ++i) {

// store the newly read score

// above the top of the stack

scanf("%d", &stack[top+1]);

// if the newly read score is distinct,

// make it the top of the stack

// note: initially top is 0 and stack[top] is 0,

// i.e. the stack contains a dummy 0 element

// that allows the code below to work

// consistently across all inputs

if (stack[top+1] != stack[top]) ++top;

}

// read m - the number of levels Alice beats

scanf("%d", &m);

// read Alice's scores and output the corresponding ranks

for (i = 0; i < m; ++i) {

// read a score

scanf("%d", &score);

// remove the top element from the stack

// until the stack only has one element

// or until the top element is greater than

// Alice's current score

while (top && score >= stack[top]) --top;

// output the current rank

printf("%d\n", top+1);

}

return 0;

}