```
#include <stdio.h>
#include <vector>

#define N 100010
#define K 200000

using namespace std;

int ancestor[100010][18]; //2^18 > 200000, the max k

int kancestor(int node, int k){
        //Break down k into its powers of two
        vector <int> powersoftwo;
        for(int i = 0, j =1; i<= 30; i++, j *= 2){
                //if(k & (1<<i)) powersoftwo.push_back(j);
                if(k & (1<<i)) powersoftwo.push_back(i);
        }

        //Go through the k-th ancestors
        int ancestork = node;
        for(auto element : powersoftwo){
                ancestork = ancestor[ancestork][element];
                if (ancestork == -1) break;
        }

        //Return the kth ancestor
        return ancestork;
}

int main(void){
        //Preliminaries
        int n, q;
        scanf("%d %d", &n, &q);

        //Scan in the pubs
        int h[n];
        for(int i = 0; i<n; i++){
                scanf("%d", &h[i]);
        }

        int mk = 1, maxkpower = 0, maxk;
        while(mk <= n){
                mk *= 2;
                maxk = mk;
                maxkpower++;
        }

        //int ancestor[n][maxkpower];

        //Find the first greater element to the right for every
element
        int firstgr[n];
        firstgr[n-1] = -1;
        ancestor[n-1][0] = -1;
```

```
        for(int i = n-2; i>=0; i--){
                int pointer = i+1;
                while(pointer != -1 && h[pointer] <= h[i]){
                        pointer = firstgr[pointer];
                }
                firstgr[i] = pointer;
                ancestor[i][0] = pointer;
                //printf("The first greater neighbour for %d is
%d\n", i, pointer);
        }


        //Now go recursively, finding the k-th element to the right
of every node that is greater than it, where k is a power of two
        int k = 1;
        /*for(int i = 0; i<n; i++){
                ancestor[i][0] = i;
        }*/
        while(k <= maxkpower){
                for(int i = 0; i<n; i++){
                        if(ancestor[i][k-1] == -1){
                                ancestor[i][k] = -1;
                                continue;
                        }
                        ancestor[i][k] = ancestor[ancestor[i][k-1]]
[k-1];
                        //printf("The greatest neighbour 2^%d hops
to the right for element %d is %d\n", k, i, ancestor[i][k]);
                }
                k++;
        }


        //Now for every node, binary search for the maximum k such
that kancestor(i, k) is a non-negative-one ancestor
        int highest_k[n];
        for(int i = 0; i<n; i++){
                int a = 0, b = mk;
                int max_ancestor = 0;
                while(a <= b){
                        int middle = (a+b)/2;
                        int middle_ancestor = kancestor(i, middle);
                        if(middle_ancestor != -1){
                                max_ancestor = middle_ancestor;
                                a = middle+1;
                        } else{
                                b = middle-1;
                        }
                }
                highest_k[i] = max_ancestor;
        }
```

```
        //Now go through the queries
        int answers[q];
        for(int query = 0; query<q; query++){
                int i, k;
                scanf("%d %d", &i, &k);

                //If the k is greater than the maximum non -1
ancestor for the node, then just set it to the highest ancestor of
that node
                if(k > highest_k[i]){
                        answers[query] = kancestor(i, highest_k[i]);

                } else{
                        answers[query] = kancestor(i, k);
                }
        }

        //Now just print out the answers;
        for(int i = 0; i<q; i++){
                printf("%d\n", answers[i]);
        }
        return 0;
}
```