```c
#include <stdio.h>

// Do binary search to find the earliest occurrence of query in
// the array arr sorted in non-decreasing order
int binSearchEarliest(int *arr, int sta, int fin, int query){
    if(fin == sta){
        if(arr[sta] == query)
            return sta;
        else
            return -1;
    }
    int center = sta + (fin - sta + 1)/2;
    if(arr[center] == query){ // Found at least one occurrences
        // Any earlier occurrences can only be to the left (sorted array)
        int prev = binSearchEarliest(arr, sta, center - 1, query);
        if(prev == -1)
            return center;
        else
            return prev;
    }else if(arr[center] < query){ // Only hope is to search the right
        return binSearchEarliest(arr, center, fin, query);
    }else if(arr[center] > query){ // Only hope is to search the left
        return binSearchEarliest(arr, sta, center - 1, query);
    }
    return -1;
}

// Find the earliest occurrence of the query element the array
// Trick is to do binary search twice on the two halves of the array :)
int find(int *arr, int len, int breakPoint, int q){
    if(breakPoint == 0) // Entire array is in sorted order
        return binSearchEarliest(arr, 0, len-1, q);
    // Do binary search in the array till the smallest element
    // This is a sorted array
    int pos = binSearchEarliest(arr, 0, breakPoint-1, q);
    if(pos >= 0) // Found it!
        return pos;
    else // Do binary search on the remaining array - that is sorted too :)
        return binSearchEarliest(arr, breakPoint, len-1, q);
}
// Find the earliest occurence of an element in the indices sta ... fin
// in the array arr that is strictly smaller than comp
int findEarliestSmaller(int *arr, int sta, int fin, int comp){
    if(fin == sta){ // Array of unit length
        if(arr[sta] < comp)
            return sta;
        else
            return -1;
    }
    int center = sta + (fin - sta + 1)/2;
    if(arr[center] < comp){ // Found at least one
        // Any smaller entries can only be to the left (sorted array)
        int prev = findEarliestSmaller(arr, sta, center - 1, comp);
        if(prev == -1)
            return center;
        else
            return prev;
    }else{ // Only hope is to search the right
        return findEarliestSmaller(arr, center, fin, comp);
    }
}

int main(){
    int n, q;
    scanf("%d", &n);
    int arr[n];
    for(int i = 0; i < n; i++)
        scanf("%d", arr + i);
    scanf("%d", &q);
```

```
    int init = arr[0];
    // If the array was rotated then there must be an element with index
    // greater than 0 which is strictly smaller than init. Find the earliest
    // such element in the array. Its index is the number of shift locations
    int pos = findEarliestSmaller(arr, 0, n-1, init);
    if(pos == -1) // init is the smallest element i.e. no rotation
        printf("0\n");
    else
        printf("%d\n", pos);
    int smallest = pos < 0 ? 0 : pos; // Location of smallest element
    printf("%d", find(arr, n, smallest, q)); // Location of q
    return 0;
}
```