Given an array with n length, and q queries
For each query, return the start position and end position of the element k. (index start from 0)
If the element doesn't exist in the array, return -1 -1

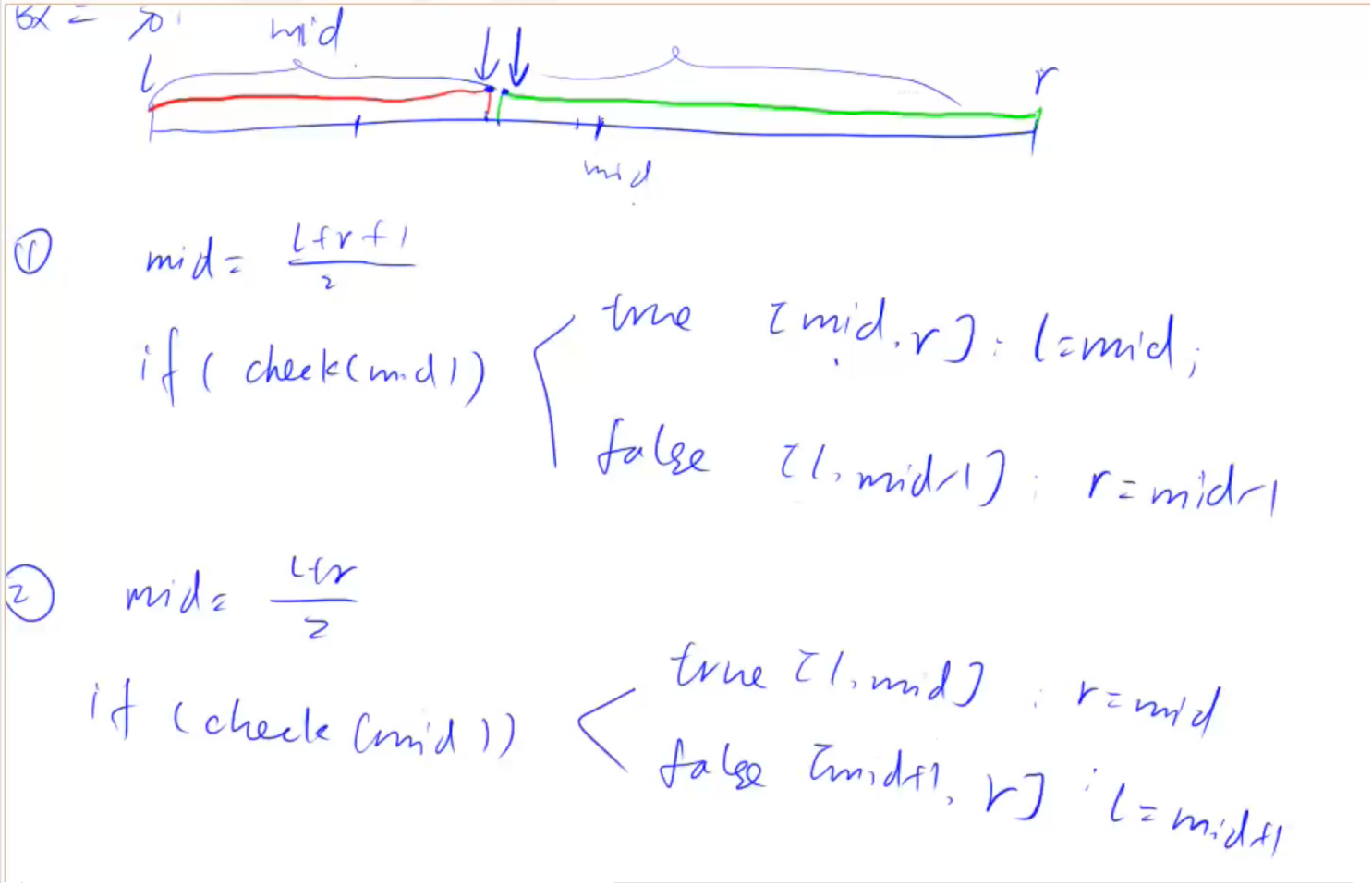
Input
n, q
n integers
query k

OUtput
start end

Range
1≤n≤100000
1≤q≤10000
1≤k≤10000

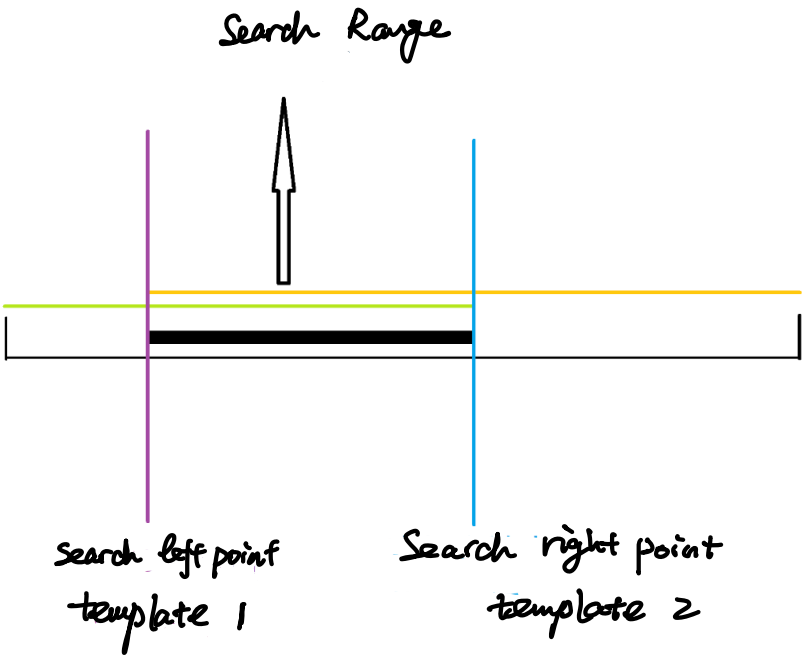
Input sample:
6 3
1 2 2 3 3 4
3
4
5

Output sample:
3 4
5 5



Template to remember

```
1  // when [l, r] divided into [l, mid] and [mid + 1, r]
2  int bsearch_1(int l, int r)
3  {
4      while (l < r)
5      {
6          int mid = l + r >> 1;
7          if (check(mid)) r = mid;    // check()判断mid是否满足性质
8          else l = mid + 1;
9      }
10     return l;
11 }
12
13 // when [l, r] divided into [l, mid - 1] and [mid, r]
14 int bsearch_2(int l, int r)
15 {
16     while (l < r)
17     {
18         int mid = l + r + 1 >> 1;
19         if (check(mid)) l = mid;
20         else r = mid - 1;
21     }
22     return l;
23 }
24
```

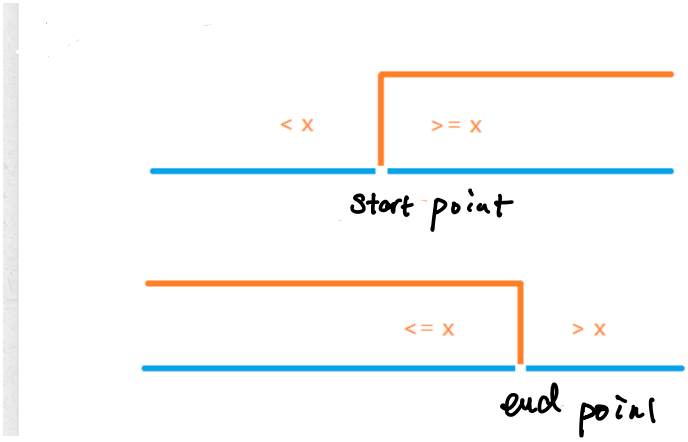


Template 1

```
while (l < r)
        {
            int mid = l + r >> 1;
            if (q[mid] >= x) r = mid;
            else l = mid + 1;
        }
```

Template 2

```
while (l < r)
            {
                int mid = l + r + 1 >> 1;
                if (q[mid] <= x) l = mid;
                else r = mid - 1;
            }
```




```cpp
#include <iostream>

using namespace std;

const int N = 100010;

int n, m;
int q[N];

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

    while (m -- )
    {
        int x;
        scanf("%d", &x);

        int l = 0, r = n - 1;
        while (l < r)
        {
            int mid = l + r >> 1;
            if (q[mid] >= x) r = mid;
            else l = mid + 1;
        }

        if (q[l] != x) cout << "-1 -1" << endl;
        else
        {
            cout << l << ' ';

            int l = 0, r = n - 1;
            while (l < r)
            {
                int mid = l + r + 1 >> 1;
                if (q[mid] <= x) l = mid;
                else r = mid - 1;
            }

            cout << l << endl;
        }
    }

    return 0;
}
```