Goal

- Today we will be taking a further look at binary search.
- We will cover the following functions:
    1. lower_bound
    2. upper_bound

Resources

Lower Bound

- https://cplusplus.com/reference/algorithm/lower_bound/
- If the element you are looking for exists in the container, then `lower_bound` will return an iterator pointing to it's first occurrence.
- If the element is not in the container, then `lower_bound` will return an iterator pointing to the position where it would be if it were inserted.
    - In other words, it will return an iterator to the first occurrence of the smallest value that is larger than the element.

Upper Bound

- https://cplusplus.com/reference/algorithm/upper_bound/
- It will return an iterator pointing to the first occurrence of the smallest value that is larger than the element.

Questions

Lower Bound

1. Given an integers n and q.
    - Followed by a list of n integers in ascending order.
    - Now you will be given q integers.
        - For each integer output the index, if the integer exists in the list given above.
        - If the integer doesn't exist in the list, print -1.
        - Your checking should be performed using lower_bound.
    - sample_input

```
10 5
1 2 3 4 5 6 7 8 9 10
7
3
11
1
10
```

    - sample_output

    ```
    6
    2
    -1
    0
    9
    ```

- Solution

```cpp
#include <bits/stdc++.h>
using namespace std;

#define all(x) (a).begin(), (a).end()

int main() {
    int n, q; cin >> n >> q;
    vector<int> a(n);
    for (int i = 0; i < n; i++)
        cin >> a[i];
    sort(all(a));
    while (q--) {
        int x; cin >> x;
        auto lb = lower_bound(all(a), x);
        if (lb == a.end() || *lb != x)
            cout << -1 << "\n";
        else
            cout << lb - a.begin() << "\n";
    }
    return 0;
}
```

2. https://cses.fi/problemset/task/1091
    - Hint: Sets and Maps have a `.lower_bound()` method. Use that!

- Solution using a Multiset

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n, m; cin >> n >> m;
    multiset<int> tickets;
    for (int i = 0; i < n; i++) {
        int x; cin >> x;
        tickets.insert(x);
    }
    for (int i = 0; i < m; i++) {
        int p; cin >> p;
        auto lb = tickets.lower_bound(p);
        if (lb != tickets.end() && *lb == p) {
            cout << *lb;
```

```
            tickets.erase(lb);
        }
        else if (lb != tickets.begin()) {
            --lb;
            cout << *lb;
            tickets.erase(lb);
        }
        else
            cout << -1;
        cout << "\n";
    }
    return 0;
}
```

- Solution using a Map
  - You can make a frequency map.
  - Decrement the frequency each time you use a ticket.
  - Erase it when the count becomes 0.

Upper Bound

1. Given an integers n and q.
   - Followed by a list of n integers in ascending order.
   - Now you will be given q integers.
     - For each integer output the index, if the integer exists in the list given above.
     - If the integer doesn't exist in the list, print -1.
     - Your checking should be performed using upper_bound.
   - sample_input

   ```
   10 5
   1 2 3 4 5 6 7 8 9 10
   7
   3
   11
   1
   10
   ```

   - sample_output

   ```
   6
   2
   -1
   0
   9
   ```

- Solution

```cpp
#include <bits/stdc++.h>
using namespace std;

#define all(x) (x).begin(), (x).end()

int main() {
    int n, q; cin >> n >> q;
    vector<int> a(n);
    for (int i = 0; i < n; i++)
        cin >> a[i];
    sort(all(a));
    for (int i = 0; i < q; i++) {
        int x; cin >> x;
        auto ub = upper_bound(all(a), x);
        if (ub == a.begin() || *(--ub) != x)
            cout << -1 << "\n";
        else
            cout << ub - a.begin() << "\n";
    }
    return 0;
}
```

2. https://cses.fi/problemset/task/1091
    - Hint: Sets and Maps have a `.upper_bound()` method. Use that!

- Solution:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n, m; cin >> n >> m;
    multiset<int> tickets;
    for (int i = 0; i < n; i++) {
        int x; cin >> x;
        tickets.insert(x);
    }
    for (int i = 0; i < m; i++) {
        int p; cin >> p;
        auto ub = tickets.upper_bound(p);
        if (ub == tickets.begin())
            cout << -1 << "\n";
        else {
            --ub;
            cout << *ub << "\n";
            tickets.erase(ub);
        }
    }
    return 0;
}
```