Goal

- Today we will be taking a further look at data structures in C++.
- Data structures for today:
  1. Stack
  2. Queue
  3. Priority Queue

Resources

Stack

- https://cplusplus.com/reference/stack/stack/
  - read up on what the following functions do:
    - push, pop
    - top
    - size, empty

Queue

- https://cplusplus.com/reference/queue/queue/
  - read up on what the following functions do:
    - push, pop
    - front
    - size, empty

Priority Queue

- https://cplusplus.com/reference/queue/priority_queue/
  - read up on what the following function do:
    - push, pop
    - top
    - size, empty
  - for custom ordering in set, maps and priority you can place a comparator in side a struct and call name it `operator`

```
struct compare {
        bool operator()(pair<int, int> &a, pair<int, int> &b) {
                if (a.first == b.first)
                        return a.second < b.second;
                return a.first > b.first;
        }
}
```

Questions

Note: Solve the questions in order.

Stack

1. Given an integer n followed by n integers.
    - Push the integers into a stack s_1 in the order of input.
    - Create a new stack s_2.
    - While the s_1 is not empty, pop the top element and push it into s_2 only if s_2 is either empty or the top element in s_2 is smaller than the current element.
        - Print the current element from s_1 regardless of whether or not it gets pushed into s_2.
    - While s_2 is not empty, pop the elements out and print them.
    - sample_input

```
10
1 5 3 2 4 8 9 12 11 3
```

    - sample_output

```
3 11 12 9 8 4 2 3 5 1
12 11 3
```

- Solution

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

int main() {
    int n; cin >> n;
    stack<int> s1;
    for (int i = 0; i < n; i++) {
        int x; cin >> x;
        s1.push(x);
    }

    stack<int> s2;
    while (!s1.empty()) {
        int curr = s1.top(); s1.pop();
        cout << curr << " ";
        if (s2.empty() || s2.top() < curr)
            s2.push(curr);
    }
    cout << "\n";

    while (!s2.empty()) {
        cout << s2.top() << " ";
        s2.pop();
    }
```

```
    return 0;
}
```

Queue

1. https://cses.fi/problemset/task/1084
    - Solve this question like before.
    - Except after sorting the vectors, transfer all the elements in order into queues.
    - After this use only the queues to assign the apartments.

- Solution

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

int main() {
    int n, m, k;
    cin >> n >> m >> k;

    vector<int> a(n), b(m);
    for (int i = 0; i < n; i++)
        cin >> a[i];
    for (int i = 0; i < m; i++)
        cin >> b[i];
    sort(a.begin(), a.end());
    sort(b.begin(), b.end());

    queue<int> aq, bq;
    for (int x : a)
        aq.push(x);
    for (int x : b)
        bq.push(x);

    int ctr = 0;
    while (!aq.empty() && !bq.empty()) {
        if (aq.front() > bq.front() + k)
            bq.pop();
        else if (aq.front() < bq.front() - k)
            aq.pop();
        else
            aq.pop(), bq.pop(), ctr++;
    }
    cout << ctr;
    return 0;
}
```

Priority Queue

1. https://cses.fi/problemset/task/1084
    - Solve this question without using vectors.
    - Directly put the elements into the priority queue.
    - By default the priority queue will return the highest elements first.

    - Using this order, solve the question.
    - Basically, you will be assigning the largest first.
        - Overall concept remains the same.

- Solution

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

int main() {
    int n, m, k;
    cin >> n >> m >> k;

    priority_queue<int> aq, bq;
    for (int i = 0; i < n; i++) {
        int x; cin >> x;
        aq.push(x);
    }
    for (int i = 0; i < m; i++) {
        int x; cin >> x;
        bq.push(x);
    }

    int ctr = 0;
    while (!aq.empty() && !bq.empty()) {
        if (aq.top() < bq.top() - k)
            bq.pop();
        else if (aq.top() > bq.top() + k)
            aq.pop();
        else
            aq.pop(), bq.pop(), ctr++;
    }
    cout << ctr;
    return 0;
}
```

2. https://cses.fi/problemset/task/1084
    - Same task as the one above, Except make the priority queue return the elements in ascending order.

- Solution

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

struct cmp {
    bool operator()(int a, int b) {
        return a < b;
    }
};

int main() {
    int n, m, k;
    cin >> n >> m >> k;
```

```
    priority_queue<int, vector<int>, cmp> aq, bq;
    for (int i = 0; i < n; i++) {
        int x; cin >> x;
        aq.push(x);
    }
    for (int i = 0; i < m; i++) {
        int x; cin >> x;
        bq.push(x);
    }

    int ctr = 0;
    while (!aq.empty() && !bq.empty()) {
        if (aq.top() > bq.top() + k)
            bq.pop();
        else if (aq.top() < bq.top() - k)
            aq.pop();
        else
            aq.pop(), bq.pop(), ctr++;
    }
    cout << ctr;
    return 0;
}
```