# Exam Preparation

Write C++ code for solving the tasks on the following pages.

Any code files that are part of the task are provided under the folder **Skeleton**.

Please follow the exact instructions on uploading the solutions for each task.

## 3. Range

You are tasked with implementing some of the methods for a **Range** class. The **Range** class represents a sorted sequence of numbers, by storing how many times each number occurs – the numbers are represented by storing the count of appearances at a corresponding index in an array.

For example, if we have the following sequence of numbers: **3 2 3 5 3 3 5**

Adding them all to a **Range** will make that range have an internal array, which looks like this:

| Index | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Value | 1 | 2 | 0 | 4 |

Here, **`index 0`** represents how many times the number **2** (the smallest in the sequence) occurs in the sequence, and **`index 3`** represents how many times the number **5** (the largest in the sequence) occurs in the sequence. If we want the numbers in sorted order, we just iterate over the array, printing out **`index + 2`** (in this case), as many times as the value of the element.

E.g., here for index **0** we print **2**, for index **1** we print **3 3**, for index **2** we don't print anything, and for index **3** we print **5 5 5**. This gives us **2 3 3 5 5 5** – the sequence sorted in ascending order.

More formally, if we call the smallest number in the range **`rangeFirst`** and the largest number in the range **`rangeLast`**, then for any value **v** such that **`rangeFirst <= v <= rangeLast`**, the element at index **`v - rangeFirst`** tells us how many times **v** was added to the range.

You are given a skeleton containing the files **`main.cpp`** and **`Range.h`**. The **`main.cpp`** file defines the **`main()`** function and solves a task in which multiple arrays are input and the one with the most occurrences of a specific number is printed out sorted to the console. To do that, it uses **Range**.

Your task is to write a **`Range.cpp`** class, which defines the implementations of:

- **`Range()`** - constructs an empty range
- **`void add(T value)`** – inserts a value into the range (resizing array if necessary)
- **`size_t getCount(T value)`** – returns the number of times value is contained in the range (expected to work in O(1) time, i.e. shouldn't depend on the size of the range)
- **`bool empty()`** – returns true the range contains no values, false otherwise
- **`Range(const Range& other);`** - copy-constructs **Range** from another **Range**
- **`Range& operator=(const Range& other);`** - copy-assigns **Range** from another **Range**
- **`~Range();`** - destructs a range
- There are also **`clear()`**, **`resize()`**,and **`getIndex()`** methods that are optional – they aren't used by any code in the skeleton, but implementing them will probably help you reuse some code

## Input

The program defined in **main.cpp** reads the following input:
One or more lines, containing arrays, ending with a line containing the string **"end"**, followed by a single integer number **Q**.

## Output

The program defined in main.cpp writes the following output:
The first array with the largest number of occurrences of the number **Q**, sorted in ascending order.

## Restrictions

Numbers in the input data are from **-100** to **100** (inclusive). Your implementation of the **Range** class should support containing any number in this range.

The total number of arrays in the input will be no more than **100**. The total number of elements in each array will be no more than **10000**.

**40%** of the test cases will contain only non-negative values in the input.

The total running time of your program should be no more than **0.4s**

The total memory allowed for use by your program is **4MB** (NOTE: "hello world" uses about **1.7MB**)

## Instructions

Submit ONLY the **Range.cpp** file, containing the implementations of all members used in the skeleton. You are NOT allowed to modify the **main.cpp** or **Range.h** files (if you do, the Judge system will just overwrite your files with its version of the skeleton).
Some of the members in the **Range.h** file are already implemented – study them and use them if you find them useful. Study the code in **main.cpp** to better understand how it uses the Range class you are implementing. This task is not just about writing code, it is also about reading and understanding C++ code you are given.

## Example I/O

| Input | Output |
|---|---|
| 1 2 3 4 5 6<br>1 2 3 3 3 6<br>1 3 3 4 5 6<br>1 2 4 5 6 7<br>end<br>3 | 1 2 3 3 3 6 |
| 10 20 3 4 5 6<br>6 3 10 3 3 -2<br>3 4 5 6 1 3<br>1 5 6 7 2 4<br>end<br>3 | -2 3 3 3 6 10 |