


```

/*
 * Sortings in STL
 */

// Sorting algorithm requires random access iterators:
// vector, deque, container array, native array

vector<int> vec = {9,1,10,2,45,3,90,4,9,5,8};

sort(vec.begin(), vec.end()); // sort with operator <
// vec: 1 2 3 4 5 8 9 9 10 45 90

bool lsb_less(int x, int y) {
    return (x%10)<(y%10);
}
sort(vec.begin(), vec.end(), lsb_less); // sort with lsb_less()
// vec: 10 90 1 2 3 4 45 5 8 9 9

// Sometime we don't need complete sorting.

// Problem #1: Finding top 5 students according to their test scores.
//
// - partial sort
vector<int> vec = {9,60,70,8,45,87,90,69,69,55,7};

partial_sort(vec.begin(), vec.begin()+5, vec.end(), greater<int>());
// vec: 90 87 70 69 69 8 9 45 60 55 7

// Overloaded:
partial_sort(vec.begin(), vec.begin()+5, vec.end());
// vec: 7 8 9 45 55 90 60 87 70 69 69

// Problem #2: Finding top 5 students according to their score, but I don't
// care their order.
vector<int> vec = {9,60,70,8,45,87,90,69,69,55,7};

nth_element(vec.begin(), vec.begin()+5, vec.end(), greater<int>());
// vec: 69 87 70 90 69 60 55 45 9 8 7

// Problem #3: Move the students whose score is less than 10 to the front
vector<int> vec = {9,60,70,8,45,87,90,69,69,55,7};

bool lessThan10(int i) {
    return (i<10);
}
partition(vec.begin(), vec.end(), lessThan10);
// vec: 8 7 9 90 69 60 55 45 70 87 69

// To preserve the original order within each partition:
stable_partition(vec.begin(), vec.end(), lessThan10);

```

```

// vec: 9 8 7 60 70 45 87 90 69 69 55
// Heap Algorithms
//
// Heap:
// 1. First element is always the largest
// 2. Add/remove takes  $O(\log(n))$  time

vector<int> vec = {9,1,10,2,45,3,90,4,9,5,8};

make_heap(vec.begin(), vec.end());
// vec: 90 45 10 9 8 3 9 4 2 5 1

// Remove the largest element:
pop_heap(vec.begin(), vec.end()); // 1. Swap vec[0] with last item vec[size-1]
// 2. Heapify [vec.begin(), vec.end()-1)
// vec: 45 9 10 4 8 3 9 1 2 5 90
vec.pop_back(); // Remove the last item (the largest one)
// vec: 45 9 10 4 8 3 9 1 2 5

// Add a new element:
vec.push_back(100);
push_heap(vec.begin(), vec.end()); // Heapify the last item in vec
// vec: 100 45 10 4 9 3 9 1 2 5 8

// Heap Sorting:
vector<int> vec = {9,1,10,2,45,3,90,4,9,5,8};
make_heap(vec.begin(), vec.end());

sort_heap(vec.begin(), vec.end());
// vec: 1 2 3 4 5 8 9 9 10 45 100
// Note: sort_heap can only work on a heap.

```