Exercise Sheet, Week 7

Question 1. Run *selection sort* on the following sequences:

$\langle 4, 2, 5, 1, 4, 6 \rangle \qquad \langle 1, 2, 3, 4, 5, 6 \rangle$

(Make sure you designate the position of the division between the left and right parts and also mark which element was selected as the smallest one in every iteration.)

Question 2. Write *selection sort* which orders an array of countries by their size. To compute the size of a country you have to use function `int size(Country x)`.

```
void sort(Country[] arr) {












}
```

Question 3. Use selection sort to sort the following cars

	Fiat 500	Citroën C1	Škoda Yeti	Formula One	Jianghuai J2	Volkswagen T2
Colour	Yellow	Red	Grey	Red	Grey	Grey
Speed	65 mph	98 mph	119 mph	233 mph	92 mph	75 mph

When comparing two cars, first compare by their colour and, if those agree, compare by their maximal speed. The order of colours is `Red < Yellow < Grey`.

Question 4. Instead of only comparing by colour and speed, we use function `cmp(Car a, Car b)` which outputs `-1` if car `a` is better than `b`, it outputs `0` if they are equally good and `1` if `b` is better than `a`. Write pseudocode for selection sort for cars, i.e. `sort(Car[] arr)`, which uses `cmp` to compare cars.

Question 5. How would the time complexity of *heap sort* change if we used, instead of binary heaps, a priority queue with operations of the following time complexities:

Operation	Time Complexity
`insert`	$\mathcal{O}(n^2)$
`deleteMin`	$\mathcal{O}(n \log n)$
`update`	$\mathcal{O}(n^2)$
`heapify`	$\mathcal{O}(n^3)$
`isEmpty`	$\mathcal{O}(1)$

(Bonus) Question 6. If calling `size` in Question 2 was too time consuming/inefficient, how would you modify the algorithm so that `size` would be called as little as possible?