1. Let $0 < \epsilon < 1$. Suppose that we have an array $A$ of $n$ items such that the first $n - n\epsilon$ items are sorted. Describe an $O(n)$ time algorithm to sort $A$.

2. Give the best-case, worst-case, best-case expected and worst-case expected running time of the following function. You can assume that the shuffle operation requires $O(n)$ time and produces each permutation of $A$ with equal probability.

Algorithms 1: Bogo($A$)

```python
1 Shuffle($A$);
2 if $A$ is sorted then
3    Return $A$;
4 else
5    Return Bogo($A$);
6 end
7 end
```

3. Consider the problem of sorting an array $A$ of $n$ elements each with multiplicity $n/k$. That is, $A$ consists of $k$ distinct elements $(y_1, y_2, \ldots, y_k)$, where each $y_i$ occurs $n/k$ times in $A$. Prove that any algorithm in the comparison model requires $\Omega(n \log k)$ comparisons to sort $A$ in the worst-case.

Note: $\forall m \geq 1, \left(\frac{m}{e}\right)^m \leq m! \leq m^m$. 