EXERCISE 1: Data Structures Review

Consider the following Symbol Table operations:

- `findMax():` Return the maximum key.
- `equals(Object other):` Check if this symbol table has exactly the same keys as `other`, regardless of the order of the keys and the values they map to.

For each of the following implementations, give the order of growth of the best- and worst-case running times of each operation in symbol tables containing n key-value pairs each. Briefly describe the algorithm of each operation.

	`findMax()`	`equals(Object other)`
Sorted Array		
Binary Search Tree		
Red-Black Binary Search Tree		
Hash Table with Chaining (assuming uniform hashing)		
Hash Table with Chaining (not assuming uniform hashing)		

EXERCISE 2: Sorting Review

(a) Which of the sorting algorithms covered in class is best for sorting an array of binary numbers? Explain.

(b) Given an arbitrary array of n integers, which of the following methods would you choose to find the minimum k numbers (in sorted order)? Explain your answer.

(i) Run k iterations of Selection Sort and return the first k elements in the array.

(ii) Run k iterations of Insertion Sort and return the first k elements in the array.

(iii) Sort using Merge Sort and return the first k elements in the array.

(iv) Quick-select the k^{th} smallest element and then Quick Sort the smallest k elements (left partition).

(c) What is the order of growth of the running time of the following solution of the problem in (b)?

```
int n = a.length;
MaxPQ<Integer> pq = new MaxPQ<Integer>();
int k = 10; // k could take any other value ≤ n.

for (int i = 0; i < n; i++) {
    pq.insert(a[i]);
    if (pq.size() > k) pq.delMax();
}

Stack<Integer> stack = new Stack<Integer>()
for (int i = 0; i < k; i++)
    stack.push(pq.delMax());
for (int i = 0; i < k; i++)
    System.out.println(stack.pop());
```