Problem 1. (*Indirect Sort*) Implement the static method `sort()` in `IndirectSort.java` that indirectly sorts `a[]` using insertion sort, ie, not by rearranging `a[]`, but by returning an array `perm[]` such that `perm[i]` is the index of the `i`th smallest entry in `a[]`.

```
$ java IndirectSort
I N D I R E C T I N S E R T I O N S O R T E X A M P L E
<ctrl-d>
A C D E E E E I I I I L M N N N O O P R R R S S T T T X
```

Problem 2. (*Merge Sorted Queues*) Implement the static method `merge()` in `MergeQueues.java` that takes two queues of sorted items as arguments and returns a queue that results from merging the queues into sorted order. Your implementation must be linear and must not alter the input queues.

```
$ java MergeQueues
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

Problem 3. (*Shannon Entropy*) A nice way of characterizing duplicates in an input is using Shannon entropy. Given an input of size N with k distinct values, for each i from 1 to k define f_i to be frequency of occurrence of the ith value and p_i to be f_i/N, the probability that the ith key value is found when a random entry of the array is sampled. The Shannon entropy of the input is calculated as

$$H = -\frac{1}{\lg N}\sum_{i=1}^{k} p_i \lg p_i.$$

Note that $H \in [0,1]$, with $H = 0$ when the N items that are all the same, and with $H = 1$ when they are all different. Implement the static method `entropy()` in `ShannonEntropy.java` that takes an array `a[]` of `Comparable` objects and returns its Shannon entropy. Your implementation must be linearithmic.

```
$ java ShannonEntropy 1000 5
H = 0.23
$ java ShannonEntropy 1000 41
H = 0.54
```

Problem 4. (*Certify Heap*) Implement the static method `maxOrderedHeap()` in `CertifyHeap.java` that takes an array `a[]` of `Comparable` objects and returns `true` if `a[]` represents a maximum-ordered heap and `false` otherwise. Your implementation must be linear.

```
$ java CertifyHeap
O T H R P S O A E I N G
<ctrl-d>
false
$ java CertifyHeap
O T S R P N O A E I H G
<ctrl-d>
true
```

Problem 5. (*Ramanujan's Taxi Redux*) Srinivasa Ramanujan was an Indian mathematician who became famous for his intuition for numbers. When the English mathematician G. H. Hardy came to visit him one day, Hardy remarked that the number of his taxi was 1729, a rather dull number. To which Ramanujan replied, "No, Hardy! It is a very interesting number. It is the smallest number expressible as the sum of two cubes in two different ways." Verify this claim by writing a program `Ramanujan.java` that takes a command-line argument N and prints out all integers less than or equal to N that can be expressed as the sum of two cubes in two different ways. In other words, find distinct positive integers i, j, k, and l such that $i^3 + j^3 = k^3 + l^3$.

In Homework 2, you used four nested `for` loops to solve the problem. A much more sophisticated implementation of the same program is using a minimum-oriented priority queue. Initialize a minimum-oriented priority queue with pairs $(1,2),(2,3),(3,4),\ldots,(i,i+1)$ such that $i < \sqrt[3]{N}$. Then, while the priority queue is nonempty, remove the pair (i,j) such that $i^3 + j^3$ is the smallest (this is easily done since we are using a minimum-oriented priority queue), print the previous pair (k,l) and current pair (i,j) if $k^3 + l^3 = i^3 + j^3 \leq N$, and then if $j < \sqrt[3]{N}$, insert the pair $(i, j+1)$ into the queue.

```
$ java Ramanujan 40000
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
13832 = 18^3 + 20^3 = 2^3 + 24^3
20683 = 19^3 + 24^3 = 10^3 + 27^3
32832 = 18^3 + 30^3 = 4^3 + 32^3
39312 = 15^3 + 33^3 = 2^3 + 34^3
```

Files to Submit

1. `IndirectSort.java`
2. `MerqeQueues.java`
3. `ShannonEntropy.java`
4. `CertifyHeap.java`
5. `Ramanujan.java`

Before you submit:

- Make sure your programs meet the input and output specifications by running the following command on the terminal:

  ```
  $ python run_tests.py -v [<problems>]
  ```

  where the optional argument `<problems>` lists the problems (`Problem1`, `Problem2`, etc.) you want to test; all the problems are tested if no argument is given.

- Make sure your programs meet the style requirements by running the following command on the terminal:

  ```
  $ check_style <program>
  ```

  where `<program>` is the `.java` file whose style you want to check.