**Problem 1.** (*Array-based Symbol Table*) Develop a symbol-table implementation `ArrayST` that uses an (unordered) array as the underlying data structure to implement the basic symbol-table API (p. 363).

```
$ java ArrayST Pluto
Mercury Venus Earth Mars Jupiter Saturn Uranus Neptune Pluto
<ctrl-d>
Mercury 1
Venus 2
Earth 3
Mars 4
Jupiter 5
Saturn 6
Uranus 7
Neptune 8
```

**Problem 2.** (*Frequency Counter*) Modify `FrequencyCounter` from the text to use `ArrayST` from above and also to print all of the values having the highest frequency of occurrence, instead of just one.

```
$ java FrequencyCounter 1
to be or not to be that is the question
<ctrl-d>
to be 2
distinct = 8
words = 10
```

**Problem 3.** (*Average GPA*) Write an `ArrayST` client called `AvgGPA` that creates a symbol table mapping letter grades to numerical scores, as in the table below, then reads from standard input a list of letter grades and computes and prints the average GPA (the average of the numbers corresponding to the grades).

| A+ | A | A- | B+ | B | B- | C+ | C | C- | D | F |
|---|---|---|---|---|---|---|---|---|---|---|
| 4.33 | 4.0 | 3.67 | 3.33 | 3.0 | 2.67 | 2.33 | 2.0 | 1.67 | 1.0 | 0.0 |

```
$ java AvgGPA
B A- A A- B A- A- B A+ B A+ A- B- B B+ B+ A C A+ F
<ctrl-d>
3.2835
```

**Problem 4.** (*Spell Checker*) Write a `SeparateChainingHastST` client called `Spell` that takes a command-line argument specifying the name of the file containing common misspellings (a line-oriented file with each comma-separated line containing a misspelled word and the correct spelling), then reads text from standard input and prints out the misspelled words in the text along with the line numbers where they occurred and their correct spellings.

```
$ java Spell misspellings.txt < data/war_and_peace.txt
unconciousness:16122 -> unconsciousness
leaded:21907 -> led
wont:39087 -> won't
wont:50591 -> won't
Ukranian:58064 -> Ukrainian
wont:59650 -> won't
conciousness:59835 -> consciousness
occuring:59928 -> occurring
```

**Problem 5.** (*Tree Traversal*) Implement the methods `preOrder()`, `inOrder()`, `postOrder()`, and `levelOrder()` in `TreeTraversal` that return the an iterable object containing nodes of a binary tree traversed in pre-, in-, post-, and level-order, respectively.

```
$ java TreeTraversal
F B G A D I C E H
<ctrl-d>
Pre-order:   F B A D C E G I H
In-order:    A B C D E F G H I
Post-order:  A C E D B H I G F
Level-order: F B G A D I C E H
```

### Files to Submit

1. `ArrayST.java`
2. `FrequencyCounter.java`
3. `AvgGPA.java`
4. `Spell.java`
5. `TreeTraversal.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.